République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Mohamed Boudiaf de M'sila

Faculté de Mathématiques et d'Informatique

Département d'informatique

Examens et corrigés-type d'optimisation combinatoire

Pour les étudiants de master IDO

Par :

Dr. Allaoua HEMMAK

2020

Préface

Ce document est un recueil des énoncés des examens session normale, examens session de rattrapage, interrogations et leurs corrigés-types que j'ai donnés à mes étudiants de master 1 et 2 de la spécialité IDO (Informatique Décisionnelle et Optimisation) dans le cadre de l'enseignement des modules d'optimisation combinatoire (Optimisation Combinatoire 1 et Optimisation Combinatoire 2 pour le master 1 et Méthodes d'optimisation pour le master 2). Ces modules font partie du cursus enseigné au département d'informatique, faculté de mathématiques et informatique, université Mohamed Boudiaf de M'sila depuis septembre 2015.

Formation Master 2 OMID
Module Méthodes Emergentes
Examen session janvier 2017

Exercice 1 (Questions de cours) (06 points)

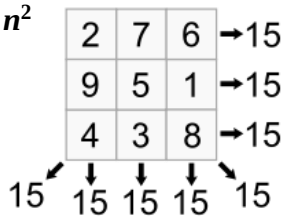
1) **Pour garantir l'approximation de la solution recherchée, toutes les métaheuristiques utilisent des paramètres d'intensification et de diversification. Compéter le tableau suivant :**

Métaheuristique	Paramètres d'intensification	Paramètres de diversification
Algorithme génétique		
Colonie de fourmis		
Colonie d'abeilles		
Essaims de particules		

2) **Expliquer pourquoi doit-on toujours préserver un compromis entre intensification et diversification ?**
3) **Donner un exemple de mesure d'approximation d'une solution approchée.**
4) **Expliciter la différence entre un algorithme génétique et un programme génétique.**
5) **Quelle métaheuristique à population de solutions est-elle considérée comme une extension de la recherche locale.**
6) **Interpréter ce que signifie "une solution" d'un problème d'optimisation pour chacun des algorithmes : PSO, ACO, BCO.**
7) **Ecrire un algorithme général d'une métaheuristique à population de solutions.**

Exercice 2 (07 pts)

Un carré magique d'ordre *n* est une matrice carrée n×n contenant les n^2 premiers entiers de 1 à n^2, de telle sorte que leurs sommes sur chaque ligne, sur chaque colonne et sur chaque diagonale principale soient égales. On appelle alors constante magique la valeur S(n) de ces sommes. On a prouvé que pour tout entier n > 2, il existe un carré magique d'ordre n et que $S(n) = (n+n^3)/2$.
Exemple : pour n=3 ; S(n)=15 (figure ci-contre).



Soit le problème : "trouver un carré magique d'ordre n".
Sachant que ce problème est en général, NP-difficile, on se propose de résoudre ce problème en utilisant un algorithme génétique.

1) **Proposer un codage pour une solution à ce problème (un individu de la population).**
2) **Quelle est la taille de l'espace de recherche ?**
3) **Ecrire une procédure qui génère une population initiale de taille K.**
4) **Ecrire les formules mathématiques que doit satisfaire une solution optimale.**
5) **Déduire une fonction fitness pour évaluer les solutions.**
6) **Proposer un opérateur de croisement et un opérateur de mutation.**

Exercice 3 (07 points)

On se propose de résoudre le problème du sac-à-dos classique *(n objets o_i ; de valeurs respectives v_i; de poids respectifs p_i , i = 1,n ; Capacité du sac: C)* en utilisant un algorithme de colonie de fourmis.

1) **Donner une représentation binaire d'une solution puis calculer le nombre de solutions candidates.**
2) **Ecrire la formulation mathématique du problème.**
3) **Combien de fourmis aura-t-on besoin ? Que devrait faire chaque agent "fourmi" et comment ?**
4) **Proposer une distance entre objets.**
5) **Ecrire l'algorithme de colonie de fourmis résolvant ce problème.**
6) **Evaluer la complexité de cet algorithme en fonction de n.**

Formation Master 2 OMID
Module Méthodes Emergentes
Examen session janvier 2017

Exercice 1 (questions de cours) (06 points)

1. ……2

Métaheuristique	Paramètres d'intensification	Paramètres de diversification
Algorithme génétique	p_c	p_m
Colonie de fourmis	α	β
Colonie d'abeilles	e	ngh
Essaims de particules	c1	c2 et c3

2. Favoriser l'intensification ⇒ convergence précoce ;
   Favoriser la diversification ⇒ mauvaise solution (aléatoire)……..…….0.5
3. ε-approximation = |solution approchée – solution optimale |/solution optimale……...0.5
4. Dans un AG, un individu est une chaine de gènes tandis que dans un PG
   un individu est un arbre représentant une fonction (un programme) en notation préfixée…0.5
5. BCO……0.5
6. PSO → une solution = une position d'une particule ;
   ACO → une solution = un chemin vers une source de nourriture ;
   BCO.→ une solution = une source de nourriture ……..……1
7. *Algorithme général d'une p-metaheuristic*……..1

```
P = P0; /* Generation of the initial population */
t = 0 ;
Repeat
        Generate(P't ); /* Generation a new population */
        Pt+1 = Select-Population(Pt ∪ P't ); /* Select new population */
        t = t + 1;
Until Stopping criteria satisfied
Output: Best solution(s) found
```

Exercice 2 (07 pts)

1. Une solution = une matrice m de n×n éléments……..……0.5
2. n^2!.....0.5
3. Génération de la population initiale de taille k……..……2

```
Pour x=1 à k
        Pour y=1 à n²
                Répéter
                        i=random(1 : n) ; j=random(1 : n) ;
                Jusqu'à m(i,j)= 0
                m(i,j) = y ;
        Fin pour
        Population (x) = m ;
Fin pour
```

4. $pour\ j = 1$ à n : $\sum_{i=1}^{n} m(i,j) = S(n)$; $pour\ i = 1$ à n : $\sum_{j=1}^{n} m(i,j) = S(n)$;
$\sum_{i=1}^{n} m(i,i) = S(n)$; $\sum_{i=1}^{n} m(i,n-i+1) = S(n)$;……..2
5. Fitness(m) = ; où $les\ S_i\ représentent\ les\ sommes\ ci-dessus$ 1
6. Operateur de mutation : permutation de 2 éléments aléatoires de la matrice m…..……0.5
   Operateur de croisement : permutation de 2 parties de 2 matrices parents
   puis correction des gènes létaux……...……0.5

Exercice 3 (07 points)

1. Une représentation binaire d'une solution = un vecteur de n bits $x_i = \begin{cases} 1\ si\ o_i\ est\ pris \\ 0\ sinon \end{cases}$ …....0.5
   Le nombre de solutions candidates = 2^n……...……..0.5
2. KSP = $\begin{cases} max \sum_{i=1}^{n} x_i . v_i \\ \sum_{i=1}^{n} x_i . p_i \leq C \end{cases}$ ……..……...……1
3. Le nombre de fourmis m doit être au moins égal à n ; chaque fourmi doit construire une solution toutes les n itérations, en effet chaque fourmi est chargée de : examiner et sélectionner les objets et mettre à jour les pistes……..……..……..…….0.5
4. $d(o_i, o_j) = \left|\frac{v_i}{p_i} - \frac{v_j}{p_j}\right|$ ……..……...……0.5
5. Algorithme de colonie de fourmis résolvant ce problème……..……..2

```
Tant que le critère d'arrêt n'est pas atteint faire
   Pour k=1 à m faire
      Choisir un objet au hasard
      Pour chaque objet non encore pris i faire
         Choisir un objet j, dans la liste des objets restants selon (P1)
      Fin Pour
      Déposer une piste sur le trajet(t) conformément à (P2)
   Fin Pour
   Évaporer les pistes selon (P3)
Fin Tant que
```

7) Evaluer la complexité de cet algorithme en fonction de n……2
   - Initialisation. → $O(n^2 + m)$;
   - Un cycle de l'algorithme.→ $O(n^2 \cdot m)$;
   - Fin du cycle et calcul des dépôts de phéromone.→ $O(n^2 \cdot m)$;
   - Evaporation des phéromones.→ $O(n^2)$;
   - Boucle de l'algorithme.→ $O(n^2 + m + Nb\_iter \cdot n^2 \cdot m)$;

   Donc l'algorithme est en $O(Nb\_iter \cdot n^2 \cdot m)$. (m = nombre de fourmis).

Formation Master 1 IDO
OPTIMISATION 2
Examen session normale mai 2018

Exercice 1 (05 pts)

On se propose de calculer les entiers a,b,c,d compris entre 1 et 20 qui satisfont: a+2b+3c+4d=50.

1) **Calculer la taille de l'espace de recherche.**
2) **Proposer une fonction fitness pour évaluer les solutions candidates de ce problème.**
3) **Ecrire un algorithme générant une solution aléatoire à ce problème.**
4) **Considérons la fonction de voisinage qui consiste à modifier un et un seul des entiers a,b,c,d d'une solution. Ecrire l'algorithme qui retourne un voisin aléatoire de la solution (a,b,c,d).**

Exercice 2 (07 pts)

Dans l'algorithme ci-contre, la fonction Rand() retourne un nombre réel r tel que $0 \leq r < 1$.

```
Float f(float x, float y)
        {  Return x * x + y * y - x * y - 2 ; } ;
Main void ()
{       Float x0 = 5 * Rand() ;
        Float y0 = 5 * Rand() ;
        Float xopt , yopt ;
        Do
        {       fopt = f(x0,y0) ;
                xopt = x0 ; yopt = y0 ;
                x0 = x0 - 0.1 + 0.2 * Rand() ;
                y0 = y0 - 0.1 + 0.2 * Rand() ; }
        While (f(x0,y0) <= fopt) ;
        Printf (xopt , yopt , fopt) ; }
```

1) **Que fait cet algorithme ?**
2) **Qu'appelle-t-on cette méthode ? Citer un avantage et un inconvénient de cette méthode.**
3) **Evaluer en fonction de x0 et y0 le nombre d'itérations de cet algorithme.**
4) **Modifier cet algorithme pour calculer le maximum de la fonction :**

$$g(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} {x_i}^2 \ \textit{pour}\ x_i \in [-2, +2]$$

en utilisant la même méthode et ses paramètres.

Exercice 1 (08 pts)

En appliquant la méthode du recuit simulé pour un certain problème d'optimisation d'une fonction *f*, on est arrivé à une température T et à la solution courante x_0 tel que $f(x_0) = 50$. Cette solution a 2 voisins x_1 et x_2 tels que $f(x_1) = 52$ et $f(x_2) = 49$. Pour chacune de ces solutions, on se propose de déterminer la probabilité pour qu'elle soit choisie pour être la prochaine solution courante ($P(x_0 \leftarrow x_1)$ et $P(x_0 \leftarrow x_2)$ pour remplacer x_0).

1) **Supposons que T= 1000 :**
   a) **Calculer ces probabilités pour un problème de maximisation.**
   b) **Même question pour un problème de minimisation.**
2) **Supposons que T= 5 :**
   c) **Calculer ces probabilités pour un problème de maximisation.**
   d) **Même question pour un problème de minimisation.**
3) **Que peut-on déduire ?**
4) **Sachant que le pallier est k et que le taux de réduction de la température est α. Exprimer en fonction de k et α le nombre d'appels de la fonction de voisinage entre T= 1000 et T= 5.**

Formation Master 1 IDO
OPTIMISATION 2
Examen session normale mai 2018

Exercice 1 (05 pts)

1. **20^4= 160000**
2. **F(a,b,c,d)= |a+2b+3c+4d-50|**
3. **a0 = int(rand()*20)+1 ; b0 = int(rand()*20)+1 ; c0 = int(rand()*20)+1 ; d0 = int(rand()*20)+1 ;**

Exercice 2 (07 pts)

1. **cet algorithme calcule le minimum de la fonction f(x)= x * x + y * y - x * y – 2 définie sur** $[0,5] \times [0,5]$**.**
2. **Méthode de la descente (Hill Climbing).**

+: facilité d'implémentation.

-: piege du min local.

3. **le nombre d'itérations ≤ f(x0,y0)**
4. **Modifier cet algorithme pour calculer le maximum de la fonction :**

$$g(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} {x_i}^2 \ \textbf{pour}\ x_i \in [-2, +2]$$

en utilisant la même méthode et ses paramètres.

```
Float f(int n, float [] x)
{       float s=0 ;
        for (int i =1; i <=n; i++)  s=s+ x[i]*x[i];
        Return s ; }
Main void ()
{       scanf n;
        float[] x0;
        for (int i =1; i <=n; i++)  x0[i] = -2+ 4* Rand() ;
        Float[] xopt;
        float fopt ;
        Do
        {       fopt = f(n,x0) ;
                for (int i =1; i <=n; i++)  xopt[i] = x0[i];
                 for (int i =1; i <=n; i++)
                      x0[i] = x0[i] - 0.1 + 0.2 * Rand() ;
        While (f(n, x0) >= fopt) ;
        Printf (xopt  , fopt) ; }
```

Formation Master 1 IDO
OPTIMISATION 2
Examen session normale mai 2018

CORRIGE-TYPE

Exercice 1 (06 pts)

1. **20^4= 160000**
   1 pt
2. **F(a,b,c,d)= |a+2b+3c+4d-50|**
   1 pt
3. **a0 = int(rand()*20)+1 ; b0 = int(rand()*20)+1 ; c0 = int(rand()*20)+1 ; d0 = int(rand()*20)+1 ; 2 pt**
4. **for (i=0;i<=3;i++) RandomNeighbor(i)= Solution(i) ;**

i = int(rand()*4) ;
do RandomNeighbor(i)= int(rand()*20)+1 while (RandomNeighbor(i)= Solution(i));
2pt

Exercice 2 (06 pts)

1) Cet algorithme calcule le minimum de la fonction: f(x)= x * x + y * y - x * y – 2 définie sur $[0,5] \times [0,5]$**. 1 pt**
2) Méthode de la descente. 0.5 pt
+: facilité d'implémentation. 0.5 pt
-: piège du min local. 0.5 pt
3) le nombre d'itérations ≤ f(x0,y0) 0.5 pt

```
4) float f(int n, float[] x)
{       float s=0 ;
        for (int i =1; i <=n; i++)  s=s+ x[i]*x[i];
        Return s ; }
Main void ()
{       scanf n;
        float[] x0;
        for (int i =1; i <=n; i++)  x0[i] = -2+ 4* Rand() ;
        Float[] xopt;
        float fopt ;
        Do
        {       fopt = f(n,x0) ;
                for (int i =1; i <=n; i++)  xopt[i] = x0[i];
                 for (int i =1; i <=n; i++)
                      x0[i] = x0[i] - 0.1 + 0.2 * Rand() ;
        While (f(n, x0) >= fopt) ;
        Printf (xopt  , fopt) ; }                    3 pt
```

Exercice 1 (08 pts)

1) T= 1000

voisins	1. Probabilités de choix en cas de maximisation	2. Probabilités de choix en cas de minimisation
a) x1 / f(x1)=52	P(x1) = 1 0.5 pt	$P(x1) = e^{-\lvert f(x1)- f(x0)\rvert/T} = e^{-0.002} = 0.998$ 1 pt
b) x2 / f(x2)=49	$P(x2) = e^{-\lvert f(x2)- f(x0)\rvert/T} = e^{-0.001} = 0.999$ 1 pt	P(x2) = 1 0.5 pt

2) T= 5

voisins	3. Probabilités de choix en cas de maximisation	4. Probabilités de choix en cas de minimisation
a) x1 / f(x1)=52	**P(x1) = 1** **0.5 pt**	**$P(x1) = e^{-\lvert f(x1)- f(x0)\rvert/T} = e^{-0.4} = 0.670$** **1 pt**
b) x2 / f(x2)=49	**$P(x2) = e^{-\lvert f(x2)- f(x0)\rvert/T} = e^{-0.2} = 0.819$** **1 pt**	**P(x2) = 1** **0.5 pt**

3) **Températures élevées $\Rightarrow$ RS $\cong$ RL**
   0.5 pt
   Températures basses $\Rightarrow$ RS $\cong$ DESCENTE
   0.5 pt
4) **Nombre de paliers = n tel que $5 = 1000*\alpha^n$ d'où $n=-\ln 200/\ln\alpha$**
   Nombre d'appels=k*nombre de paliers= $- k* \ln 200/\ln\alpha$
   1 pt

Formation Master 2 IDO
Module Méthodes Emergentes
Examen session janvier 2018

Exercice 1 (05 points)

Considérons un algorithme génétique appliqué à une population de 8 entiers (individus). Supposons que la fitness est $f(x) = x^2$ et qu'à la génération t la composition de la population est comme suit :

1	1	2	2	2	3	3	3

1) **Ecrire l'algorithme de sélection par roulette.**
2) **Calculer les probabilités respectives de sélectionner chacun des individus 1 , 2 et 3 en utilisant la méthode de la roulette.**
3) **Sachant que chaque individu est codé en binaire sur 4 bits, en appliquant un croisement en un point et une mutation qui change le dernier bit à droite, quels seront les individus pouvant constituer la population de la génération t+1 ?**

Exercice 2 (08 points)

On se propose d'appliquer l'optimisation par essaim de particules (PSO) pour rechercher le maximum de la fonction *f* à une variable réelle x définie par : $f(x) = -x^2 + 4x$ avec $x \in [-5, +5]$.

1) **En partant, dans la phase d'initialisation, par les trois particules $x_1 = 2$; $x_2 = 4$; $x_3 = -1$ avec les vélocités initiales $v_1 = v_2 = v_3 = 0$ et en utilisant une inertie $c_1 = 0.8$ et des accélérations $c_2 = c_3 = 2$.**
   a) **Calculer la meilleure position $pbest_i$ i=1,3 pour chaque particule.**
   b) **Calculer la meilleure position globale de l'essaim gbest.**
   c) **Calculer les nouvelles valeurs x^1_i ; v^1_i ; $pbest^1_i$; i=1,3 et $gbest^1$ à la première itération.**
2) **Afin d'implémenter cet algorithme, on prendra n particules. et en utilisant les mêmes paramètres qu'en (1).**
   a) **Ecrire l'algorithme permettant de générer une population initiale de n particules aléatoires x_i avec des vélocités aléatoires v_i , i= 1,n et de calculer les $pbest_i$ et gbest.**
   b) **Evaluer la complexité de l'algorithme d'initialisation.**
   c) **Ecrire l'algorithme PSO intégral pour ce problème puis évaluer sa complexité.**

Exercice 3 (07 pts)

Une entreprise fabrique deux types de produits A et B.
La réalisation d'une unité de A demande 30 DA de matière première et 120 DA de main d'œuvre.
La réalisation d'une unité de B demande 60 DA de matière première et 80 DA de main d'œuvre.
Les profits réalisés sont de 50 DA par unité de A, et de 40 DA par unité de B.
Les contraintes économiques font que :

- ✓ **La dépense journalière en matière première ne doit pas dépasser 580 DA.**
- ✓ **La dépense journalière en main d'œuvre ne doit pas dépasser 1250 DA.**

On désire trouver le plan de production journalière (c.-à-d. les nombres d'unités de A et B à fabriquer par jour) qui maximise le profit journalier de l'entreprise.

1) **Donner une formulation mathématique à ce problème.**
2) **En appliquant l'optimisation par colonie d'abeilles :**
   a) **Ecrire l'algorithme d'initialisation d'une population de 10 scouts.**
   b) **Définir une fonction de voisinage de votre choix. Donner un exemple.**
   c) **Ecrire un algorithme permettant de retourner le meilleur des k voisins d'une solution.**

Formation Master 2 IDO
Module Méthodes Emergentes
Corrigé type Examen session janvier 2018

Exercice 1 (04 points)

1) int roulette() ; (01.5 pt)

```
{       r = sumfitness*rand() ;
        s = 0;
        i=0;
        while (s < r)
        {       i++;
                s = s + fitness[i] ; }
        return i ; }
```

2) probabilités de sélection (02 pt)

x_i	$f(x_i)$	$n_i*f(x_i)$	Probabilité(x_i)
1	1	2	2/41
2	4	12	14/41
3	9	27	27/41
sum		41	1

3) quels que soient le croisement et la mutation des entiers 0011 0001 0010, les deux bits de gauche restent 00, donc les individus de t+1 seront 0,1,2 ou 3. (0.5 pt)

Exercice 2 (11 points)

1) **a) $Pbest_1 = x_1 = 2$; $Pbest_2 = x_2 = 4$; $Pbest_3 = x_3 = -1$ (1 pt)**

b) $f(x_1) = 4; f(x_2) = 0; f(x_3) = -3;$ **gbest = $x_1 = 2$. (1 pt)**

c) $v^1_1 = c_1 v^0_1 + c_2(Pbest_1 - x_1) + c_3(gbest - x_1) = 0$; $x^1_1 = x^0_1 + v^0_1 = 2$; $f(x_1) = 4$; $pbest^1_1 = 2$; **(1.5 pt)**

$v^1_2 = c_1 v^0_2 + c_2(Pbest_2 - x_2) + c_3(gbest - x_2) = 4$; $x^1_2 = x^0_2 + v^0_2 = -2$; $f(x_2) = -12$; $pbest^1_2 = 4$; **(1.5 pt)**

$v^1_3 = c_1 v^0_3 + c_2(Pbest_3 - x_3) + c_3(gbest - x_3) = 6$; $x^1_1 = x^0_3 + v^0_3 = 5$; $f(x_3) = -5$; $pbest^1_3 = 2$; **(1.5 pt)**

gbest = $x_1 = 2$. (0.5pt)

2) a) void initialize() ; (02 pt)

```
{ gbest = -5 ;
  for (i=1; i<=n ; i++)
{       x[i]= -5+rand()*10 ;
        Pbest[i]=x[i] ;
        v[i]= rand()*10 ;
        if (f(pbest[i]) > f(gbest))
                gbest =pbest[i] ;
}}
```

2) b) Complexité = O(n)

```
2) c) void PSO() ; (02 pt)
{ initialize() ;
  for (iter=1; iter<=maxiter ; iter++)
{    for (i=1; i<=n ; i++)
       {v[i] =c1* v[i]  + c2*(Pbest[i] – x[i]) + + c2*(gbest – x[i]) ;
        x[i] = x[i] + v[i] ;
        if (f(x[i]) > f(pbest[i]))
              { pbest[i] = x[i] ;
                if (f(pbest[i]) > f(gbest))
                        gbest =pbest[i] ;
}}}}
Complexité = O(maxiter*n)
```

Exercice 3 (05 pts)

1) Soient x_1 et x_2 les quantités respectives de A et de B produites par jour. (02 pt)

Le problème s'écrit : $\begin{cases} \max(50x_1 + 40x_2); x_1, x_2 \in N \\ 30x_1 + 60x_2 \leq 580 \\ 120x_1 + 80x_2 \leq 1250 \end{cases}$

2) a) on remarque que $x_1 \leq 10$ et $x_2 \leq 10$ donc x1 et x2 seront des entiers aléatoires de l'intervalle [0,10]

```
void initialize() ; (01 pt)
{ for (i=1; i<=10 ; i++)
{        x[1, i] = int(10*rand()) ;
         x[2, i] = int(10*rand()) ;
}}  (1pt)
```

```
c) int bestneighbor(int V[][]) ; (01 pt)
{ bN=1;
for (i=2; i<=k ; i++)
   if (50*V[1, i]+40*V[2, i] > 50*V[1, bN]+40*V[2, bN])
         bN = i ;
return bN ; }  (1pt)
```

b) $V(x_1,x_2)$= {(x_1+1,x_2), (x_1-1,x_2) , (x_1,x_2+1) , (x_1,x_2-1) } **(0.5 pt)**

exemple V(4,7) = {(5,7), (3,7) , (4,8) , (4,6) } **(0.5 pt)**

Formation Master 1 IDO
Optimisation Combinatoire 2
Examen 1 / session normale juin 2019

Exercice 1 (07 pts)

En appliquant la méthode du recuit simulé pour un certain problème, on est arrivé à une itération où la température T =100 et la valeur de la fonction objectif de la solution courante est 30. Cette solution admet trois voisins x1, x2, x3 dont la valeur de la fonction objectif est 32, 25 et 29 respectivement. Pour chacune de ces solutions, on se propose de déterminer la probabilité pour qu'elle soit choisie pour être la prochaine solution courante.

1) **Déterminer ces probabilités pour un problème de maximisation puis pour un problème de minimisation.**
2) **Expliquer le comportement de l'algorithme du recuit simulé lorsque la température est élevée et lorsque la température est basse.**
3) **Que justifie ce comportement ?**

Exercice 2 (07 pts)

Dans l'algorithme ci-contre, la fonction Rand() retourne un nombre réel aléatoire compris entre 0 et 1.

1. **Que fait cet algorithme ?**
2. **Qu'appelle-t-on cette méthode ? de quelle variante s'agit-il ?**
3. **Citer un avantage et un inconvénient de cette méthode.**
4. **Modifier cet algorithme pour calculer le maximum de la fonction :**

$$h(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} x_i^{-1} \ pour \ x_i \in [-5, +5]$$

en utilisant la même méthode et ses paramètres, puis évaluer sa complexité en fonction de n.

```
float f(float x, float y)
     {  return x * x - y * y + 2*x * y + 1 ; }
main void ()
{  float x0 = 10 * Rand() ;
   float y0 = 10 * Rand() ;
   float x1 , y1 ;
   maxiter=1000 ;
   for(int iter=1; maxiter; iter++)
    { count=0;
      do
              {  x1 = x0 - 0.05 + 0.1 * Rand() ;
                 y1 = y0 - 0.05 + 0.1 * Rand() ;
                  count++; }
     while (f(x1,y1) >= f(x0,y0)  && count<10 )
     if (count<10)
          {x0=x1;y0=y1; }
      else
         break; }
      s.o.p (x0 , y0 , f(x0,y0)) ; }
```

Exercice 3 (06 pts)

On se propose de calculer les entiers a,b,c compris entre 1 et 30 qui satisfont: a+2b+3c =40.

1. **Quel est le type de ce problème d'optimisation ?**
2. **Calculer la taille de l'espace de recherche.**
3. **Proposer une fonction fitness (objectif) pour évaluer les solutions candidates de ce problème.**
4. **Ecrire un algorithme générant une solution aléatoire à ce problème.**
5. **Considérons la fonction de voisinage qui consiste à modifier un et un seul des entiers a,b,c d'une solution. Ecrire l'algorithme qui retourne un voisin aléatoire de la solution (a,b,c).**

Formation Master 1 IDO
Optimisation Combinatoire 2
Examen 1 / session normale juin 2019

Exercice 1 (06 pts)

1)

voisins	Probabilités en cas de minimisation	Probabilités en cas de maximisation
x1 / f(x1)=32	P(x1) = $e^{-(f(x1)- f(x0))/t}$= $e^{-0.02}$= 0.9801 pt	P(x1) = 1……...0.5 pt
x2 / f(x2)=25	P(x2) = 1……. 0.5 pt	P(x1) = $e^{-(f(x0)- f(x2))/t}$= $e^{-0.05}$= 0.951.1 pt
x3 / f(x3)=29	P(x3) = 1……. 0.5 pt	P(x3) = $e^{-(f(x0)- f(x3))/t}$= $e^{-0.01}$ = 0.990.1 pt

2) **Températures élevées ⇒ RS ≅ RLA…….0.5 pt**
Températures basses ⇒ RS ≅ DESCENTE…….0.5 pt
3) **Températures élevées ⇒ diversification……..……..0.25 pt**
Températures basses ⇒ intensification……..……...0.25 pt

Exercice 2 (09 pts)

1. **Cet algorithme recherche le minimum (approché) de la fonction réelle f à deux variables réelles x et y définie sur $[0,10]^2$ par $f(x,y)= x^2 – y^2 + 2xy + 1$.....2 pt**
2. **Méthode de la descente, first deep…..1 pt**
3. **+ intuitif, facile à implémenter, peu de paramètres …….0.5 pt**

- piège du minimum local……..0.5 pt
4. Algorithme modifié ci-contre…….3 pt
Complexité de l'algorithme :
La fonction : O(n)
Initialisation : O(n)
Corps de l'algorithme : Trois boucles imbriquées en O(10*n*maxiter)
Donc l'algorithme est en O(10^4*n)……..2 pt

```
float f(float[] x )
      {  float s = 0;
         for (int i=0; i<=n-1; i++)
               s = s + 1/x[i];
          return s ; }
main void ()
{  float x0[]; float x1[];
   for (int i=0; i<=n-1; i++)
         x0[i] = 10 * Rand() ;
   maxiter=1000 ;
   for(int iter=1; maxiter; iter++)
    { count=0;
      do
          {  for (int i=0; i<=n-1; i++)
                x1[i]  = x0[i]  - 0.05 + 0.1 * Rand() ;
                count++; }
      while (f(x1) >= f(x0)  && count<10 )
      if (count<10)
              for (int i=0; i<=n-1; i++)
                   x0[i]  = x1[i] ;
       else
          break;  }
      s.o.p (x0 , f(x0)) ; }
// on peut rajouter un ε>0 aux variables pour
éviter la division par 0.
```

Exercice 3 (05 pts)

1. **Discret…… 0.5 pt**
2. **Taille de l'espace de recherche = 30*30*30 = 27000……0.5 pt**
3. **Fonction fitness (objectif) : min f(a,b,c) = │a+2b+3c - 40│…….. 1 pt**
4. **x0[0] = int(30*rand())+1 ; x0[1] = int(30*rand())+1 ; x0[2] = int(30*rand())+1 …….1.5 pt**
5.

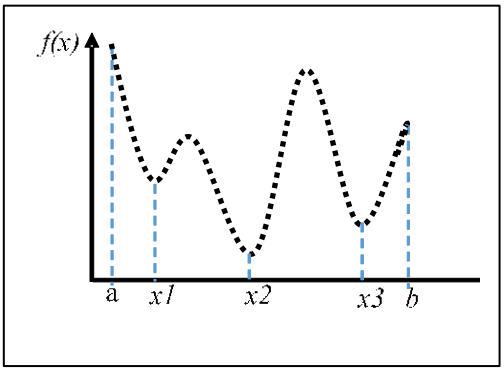
```
Int[] x1(int[] x0)
      {  x1=x0 ;
         int i=int(3*rand()) ;
         x1[i] = int(30*rand())+1 ;
        return x1 ; }
```

……1.5 pt

Formation Master 2 IDO
Module Méthodes d'optimisation
Examen session janvier 2019

Exercice 1 (08 points)

On se propose de **minimiser** une fonction numérique *f* d'une variable **entière** *x* dont la représentation graphique est illustrée dans la figure ci-contre.



1) Ecrire la formulation mathématique de ce problème d'optimisation en y précisant **l'espace de recherche** et **les contraintes**.
2) Evaluer la taille de l'espace de recherche.
3) Quel est le type de ce problème d'optimisation ?
4) Que représentent les entiers x_1, x_2 et x_3 ?
5) En résolvant ce problème par un algorithme d'optimisation par colonie d'abeilles, on a récapitulé les fréquences des solutions x_1, x_2 et x_3 de 100 exécutions de cet algorithme pour différentes configurations des paramètres n_1 et n_2 dans le tableau suivant : (n_1: nombre de scouts pour les sources riches en nectar et n_2 : nombre de scouts pour les sources pauvres en nectar).

Paramètres		x_1	x_2	x_3
1	$n_2 = 10*n_1$	32	33	35
2	$n_1 = 10*n_2$	16	66	18
3	$n_1 = 2*n_2$	3	93	4

   a) Quelle est la meilleure configuration des paramètres ? justifier.
   b) Interpréter les deux autres configurations.

Exercice 2 (08 points)

La fonction de Rastrigin est une fonction numérique *f* à n variables (de R^n vers R) définie par :

$$f(x) = f(x_1, x_2, \dots, x_n) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right) + 10n \; ; x_i \in [-5.12, +5.12]$$

Cette fonction admet un très grand nombre de minima locaux et un seul minimum global f(0,0,…,0)=0.

On se propose de calculer un **minimum** approché à cette fonction en utilisant l'optimisation par essaim de particules.

1) Définir ce que serait (**donner un exemple pour chaque cas**) :
   a) Une position d'une particule.
   b) Une vélocité d'une particule.
   c) Une somme d'une position et d'une vélocité.
   d) Une différence entre deux positions.
   e) Un produit d'un scalaire par une vélocité.
2) Ecrire l'algorithme PSO correspondant puis évaluer sa complexité.

Exercice 3 (04 points)

Le tableau suivant fournit les valeurs de deux critères (objectifs) à **maximiser** pour 5 solutions candidates :

solution	f_1	f_2
a	5	15
b	18	4
c	10	10
d	9	7
e	5	18

1) Représenter l'espace des objectifs pour ce problème.
2) Calculer le front Pareto optimal de ce problème.
3) Calculer la solution optimale par une méthode de la somme pondérée en utilisant les deux coefficients $w_1 = w_2 = 0.5$.
4) Que peut-on déduire ?

Formation Master 2 IDO
Module Méthodes d'optimisation
Examen session janvier 2019 – Corrigé-type

Exercice 1 (05 points)

1. Formulation mathématique
$\begin{cases} min_{x\in[a,b]} f(x) \\ x \in N \end{cases}$ ……..……1
2. Taille de l'espace de recherche = $\left|\,[a,b]\,\right| = b - a + 1$ ……..……...……0.5
3. Type : problème d'optimisation combinatoire (discret) ……..……..……0.5
4. x_1 et x_3 : minimiseurs locaux x_2 : minimiseur global ……...……..……1
5. Interprétation des résultats

a. Meilleure configuration = 3. Dans la quasi-totalité des cas, on obtient le minimum global en x_2 …1

b. Dans le premier cas, on a favorisé la diversification au détriment de l'intensification,
la recherche est alors pseudo aléatoire et les solutions sont à égalité de chances……..…0.5
Dans le deuxième cas, on a favorisé l'intensification au détriment de la diversification.
Le piège du minimum local persiste…….……..……..…....……0.5

Exercice 2 (11 points)

1. formulation :

a. Une position = un vecteur $x = (x_1, x_2, \dots, x_n)$; $x_i \in [-5.12, +5.12]$; ex: n=3 ; x=(1,0,2).1

b. Une vélocité = un vecteur $v = (v_1, v_2, \dots, v_n)$; $v_i \in [0,10.24]$; ex: v=(1,1,1)
…….……..…1

c. Une somme d'une position x et d'une vélocité v = une position y tel que y_i=max(x_i+v_i,5.12).
Ex: (1,0,2)+(1,1,1)=(1,0,2) ……...……1

d. Une différence entre deux positions x et y = vélocité v tel que $v_i = \left|x_i - y_i\right|$
Ex: (1,0,2)-(1,-1,3)=(0,1,1) ……..……...……1

e. Un produit d'un scalaire c par une vélocité v = une vélocité w tel que w_i = max($c.v_i$, 10.24)

```
// Initialisation
Int n,m,maxiter ;
Scanf n ;  // dimension de f
Scanf m ; // nombre de particules
Scanf maxiter ; // nombre d'itérations
Int i,j ;
Float x[][] , v[][] , pbest[][] ;
For (i = 0; i<m; i++)
  {   For (j = 0; j<n; j++)
        {  x[i,j] = -5.12+10.24*rand(0,1) ;
           v[i,j] = 10.24*rand(0,1) ; }
      pbest[i] = x[i] ; } }……...…2
```
complexité O(n*m)…….……..0.5

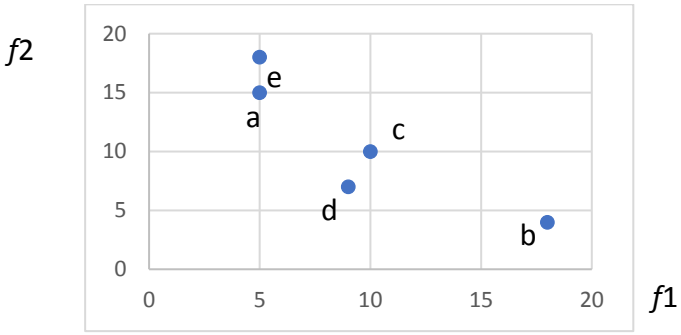
Ex: 2.(1,0,1)=(2,0,2) ……..……1

2. Algorithme PSO

```
// main
Const infini= 1000000 ;
Float fopt = infini , gbest ;
For (iter = 0; iter < maxiter ; iter++)
   {  For (i = 0; i<m; i++)
         if (f(pbest[i]) < fopt)
              { gbest = pbest[i]  ; fopt= f(gbest) ; }
       For (i = 0; i<m; i++)
          {  For (j = 0; j<n; j++)
               { v[i,j] = c1* v[i,j] +c2*abs( x[i,j]-pbest[i,j])
                                    +c3*abs( x[i,j]- gbest);
                  x[i,j] = x[i,j]+ v[i,j];                    }
             if (f(x[i]) < f(pbest[i]))  pbest[i] = x[i] ; } }
cout gbest, fopt ; ……..……3
complexité O(n*m*maxiter)…….……0.5
```

Exercice 3 (06 points)

1. Espace des objectifs : ……...……2



2. Front Pareto optimal : a est dominé par e ; d est dominé par c ; e,c et b ne sont dominés par aucune solution ; donc le front Pareto optimal est {e,c,b}. ……..……..1.5
3. Somme pondérée

solution	f_1	f_2	*0.5f1+0.5f2*
a	5	15	10
b	18	4	11
c	10	10	10
d	9	7	8
e	5	18	**11.5**

⇒ Solution optimale = e ……...2

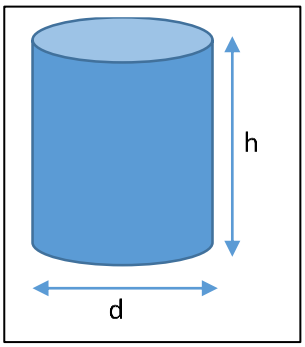
5) La solution c est mieux équilibrée que e, c'est l'inconvénient de la somme pondérée….0.5

Formation Master 2 IDO
Module Méthodes d'optimisation
Examen session janvier 20

Exercice 1. Questions de cours (07 points)

1) Citer les trois différences entre une solution dans un algorithme génétique et une solution dans un algorithme de programmation génétique.
2) Expliquer brièvement les trois approches d'initialisation dans un algorithme de programmation génétique.
3) Que signifie "*une méthode constructive*". citer un exemple vu au cours.

Exercice 2 (07 points)

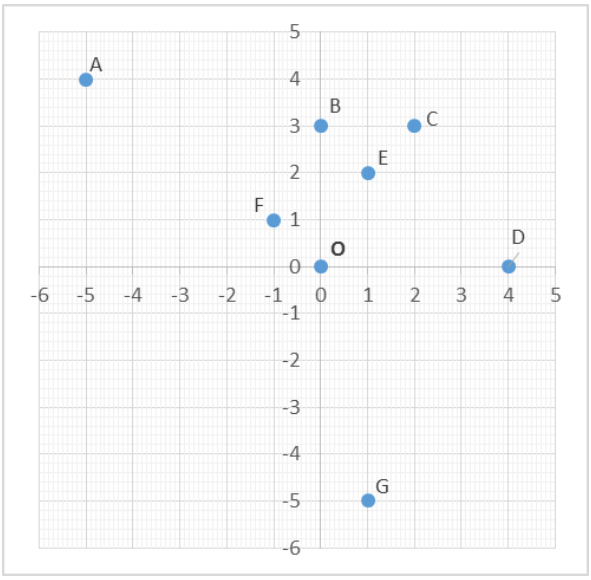


On désire fabriquer des cylindres métalliques d'une capacité au moins égale à 300 litres et d'une quantité de métal minimale.
La quantité du métal nécessaire à la fabrication d'un cylindre est égale à sa surface et elle est donnée par la formule . Son volume est donné par la formule où h est la hauteur du cylindre et d est le diamètre de sa section tels que d *et* h *sont des réels et* $0 < d \leq 16$ *et* $0 < h \leq 16$.
L'objet de cet exercice est de déterminer les valeurs optimales des dimensions d et h qui satisfont ces contraintes et minimisent la quantité de métal.

1) Ecrire la formulation mathématique de ce problème sous forme d'un problème d'optimisation.
2) Quel est le type de ce problème d'optimisation ?
3) Expliciter l'espace de recherche de ce problème.
4) Afin de résoudre ce problème à l'aide d'un algorithme génétique, on code une solution par un couple de réels (d,h).

a) Ecrire l'algorithme d'initialisation d'une population de m individus et évaluer sa complexité.
b) Ecrire l'algorithme de croisement de probabilité pc qui consiste à permuter les hauteurs de deux solutions successives de la population puis évaluer sa complexité.

Exercice 3 (06 points)



On dispose de n points $A_i(x_i, y_i)$ dans un plan euclidien muni d'un repère orthonormé d'origine O(0,0).
On se propose de déterminer les k points (parmi n) les plus proches de l'origine O (k≤n). Dans l'exemple illustratif ci-contre, avec n=7 et k=3, la solution optimale du problème est : s*={B(0,3) ; E(1,2) ; F(-1,1)} avec : $d(O,B) = 3$; $d(O,E) = \sqrt{5}$; $d(O,F) = \sqrt{2}$. (Rappel : la distance entre l'origine O(0,0) et le point A(x ;y) est d(O,A) = $\sqrt{x^2 + y^2}$).

1) Donner la fonction objectif à minimiser dans ce problème.

2) Quel est le type de ce problème d'optimisation ?
3) Evaluer la taille de l'espace de recherche.
4) On se propose de résoudre ce problème à l'aide d'un algorithme de colonies de fourmis.
    a) Quelles sont les composantes d'une solution ?
    b) D'où les fourmis devraient-elles débuter la construction des solutions ?
    c) Définir une fonction de distance entre deux composantes.
    d) Sur quelles arrêtes les fourmis devraient-elles déposer de la phéromone ?

IDO Master 2

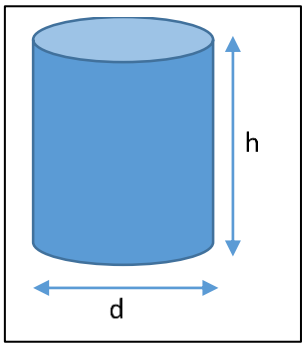
Optimization Methods module

January 2020 session exam

Exercise 1. Theoretical questions (07 points)

1) List the three differences between a solution in a genetic algorithm and a solution in a genetic programming algorithm.
2) Briefly explain the three initialization approaches in a genetic programming algorithm.
3) What does "a constructive method" mean? Cite an example seen in class.

Exercise 2 (08 points)



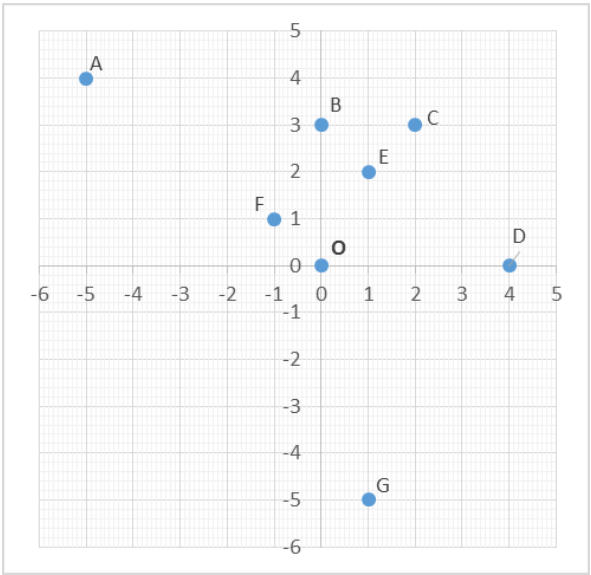
We want to manufacture metal cylinders with a capacity of at least 300 liters and a minimum quantity of metal.
The quantity of metal necessary for the manufacture of a cylinder is equal to its surface and is given by the formula . Its volume is given by the formula where h is the height of the cylinder and d is the diameter of its section such that d and h are real numbers and $0 < d \leq 16\ et\ 0 < h \leq 16$. The purpose of this exercise is to determine the optimal values of the dimensions d and h which satisfy these constraints and minimize the amount of metal.

1) Write the mathematical formulation of this problem as an optimization problem.
2) What is the type of this optimization problem?
3) Explain the search space for this problem.
4) In order to solve this problem using a genetic algorithm, we code a solution by a couple of reals (d, h).

a) Write the initialization algorithm for a population of m individuals and then compute its complexity.
b) Write the crossover algorithm, which consists in permuting the heights of two successive solutions of the population, and then compute its complexity.

Exercice 3 (06 points)

Given n points $A_i(x_i, y_i)$ in a Euclidean plane provided with an orthonormal reference frame of origin O (0,0).
We propose to determine the k points (among n) closest to the origin O (k≤n). In the illustrative example opposite, with n = 7 and k = 3, the optimal solution of the problem is: s * = {B (0.3); E (1,2); F (-1,1)} with::
$d(O,B) = 3\ ; d(O,E) = \sqrt{5}\ ;\ d(O,F) = \sqrt{2}$.
(Reminder: the distance between the origin O (0,0) and the point A (x; y) is d(O,A) = $\sqrt{x^2 + y^2}$).



5) Give the objective function to minimize in this problem.
6) What is the type of this optimization problem?
7) Evaluate the size of the search space.

8) We propose to solve this problem using an ant colony algorithm.
    a) What are the components of a solution?
    b) Define a distance function between two components.
    c) Where should ants start building a solution?
    d) On which edges should ants deposit pheromone?

Master 2 ido
Optimization Methods module
January 2020 session exam

Exercise 1. Course questions (05.5 points)

1) in GA, a solution is a fixed length linear string. In GP, a solution is a variable length tree program.

2)

	intensification	diversification
GA	pc	pm
ACO	alpha	beta
BCO	n1 & n2	n-m

3) A constructive method = method where the solution is built component by component. Example: ACO.

Exercise 2 (08 points)

1. Mathematical formulation:
2. type = Continuous optimization problem.
3. Search space = $[0,16]^2 \subseteq R^2$.
4.

c) Initialization algorithm

```
void initialization()
{
for(i=0 ; i≤ m-1 ; i++)
{
    population[i][0]= 16*rand();
    population[i][1]= 16*rand();
}
}
```

Complexité O(2m)

d) Crossover algorithm

```
void crossover()
{
for(i=0 ; i≤ m-2 ; i+=2)
{
  r=rand();
   if (r≤pc)
    {
       x = population[i][1];
       population[i][1] = population[i+1][1];
       population[i+1][1]=x;
    }
}
}
```

Complexity = O(2m)

Exercice 3 (06.5 points)

1. Objective $f(s) = \sum_{A_i \in S} d(O, A_i) = \sum_{A_i \in S} \sqrt{{x_i}^2 + {y_i}^2}$.
2. Type = Combinatorial optimization problem.
3. Size of the search space = C_n^k.
4.
   a. Components of a solution s = k points $A_i(x_i, y_i)$ among n.
   b. Distance between two components $A_i(x_i, y_i)$ and $A_j(x_j, y_j)$= d(O,$A_j(x_j, y_j)$)= $\sqrt{{x_j}^2 + {y_j}^2}$
   c. All ants should start from the origin O.
   d. Then ant which build the solution s should deposit the pheromone on the edge O$A_i if A_i \in s$.

CORRIGE-TYPE / EXAMEN AA / SESSION JANVIER 2017

Questions de cours (8pts)

a) La recherche dichotomique se fait en k étapes tel que k ≤ **logn** (en majorant n par 2^n)..…1pt

b) Le tri par insertion est en **$\Theta(n^2)$** dans les pires cas……...……1pt

c) Le nombre de solutions candidates pour le KSP simple est = le nombre d'entiers de n bits = **2^n** ……1pt

d) La récursivité imbriquée consiste à faire un appel récursif à l'intérieur d'un autre……1pt

e) O décrit une borne sup de la complexité d'un algo (celle des pires cas), tandis que Omega décrit une borne inf de la complexité (celle des meilleurs cas)……..1pt

f) La complexité de cet algo est défini par $T(n)= 8T(n/2)+n^2$.
$T(n) = 8[8T(n/4)+(n/2)^2]+n^2$ (en substituant T(n/2)).
$= n^2+2n^2+8^2T(n/2^2)$
$= 2^0n^2 + 2^1n^2 + 2^2n^2 + 2^3n^2 + \ldots + 2^{logn-1}n^2 + 8^{logn}T(1)$ en majorant n par 2^n.
$= (2^{logn} -1)\, n^2 + 2^{3logn} *T(1)$
$= (n -1)\, n^2 + n^3 *T(1)$
= **$O(n^3)$** (T(1) est bien évidemment constant)……..……1.5pt

g) Le nombre d'itérations de la boucle interne est toujours pair et = i =2.k.
Le nombre d'itérations de la boucle externe est N/2 si N est pair et (N-1)/2 si N est impair
Le nombre d'affichages de * est alors $\begin{cases} \sum_{k=1}^{\frac{N}{2}} 2.k = 2.\frac{\left(\frac{N}{2}\right)\left(\frac{N}{2}+1\right)}{2} = \frac{N^2+2N}{2} \\ \sum_{k=1}^{\frac{N-1}{2}} 2.k = 2.\frac{\left(\frac{N-1}{2}\right)\left(\frac{N-1}{2}+1\right)}{2} = \frac{N^2-1}{2} \end{cases}$ ……..1.5pt

Exercice 1

Il faut passer le résultat sum en paramètre pour éviter la remontée…….……3pts

```
Int SumTab( int tab[] , int k, int sum ) ;
  {
     if (k = 1 )
        return tab[k];
     else
        {
          sum = sum + tab[k-1] ;
          return SumTab(tab, k-1,sum) ;
        }
  }
```

Appel initial SumTab(tab, n , tab[n]) ;

Exercice 2

a) Le nombre de multiplications de cet algo est = $\sum_{i=0}^{n}\left(\sum_{j=1}^{i} 1 + 1\right) = \sum_{i=0}^{n}(i+1) = \frac{(n+1)(n+2)}{2}$.
Il est donc en **$O(n^2)$**…….……2pt

b) Son inconvénient est qu'il recalcule les mêmes puissances plusieurs fois (boucle interne)1pt

c) En posant $P(x)=a_0+x(a_1+x(a_2+x(a_3+\ldots\ldots x(a_{n-1} + xa_n)\ldots)$ on en élabore l'algo suivant en O(n):

```
Double FastEvalPoly(int n,double a[],double x);
   {
        Double s= a[n] ;
        For (int i=n-1 ; i=0;  i--)
           s = a[i]+x*s ;
        Return s :
    }
```

……2pts

d) Version récursive

```
Double Eval(int k,double a[],double x, double poly);
   {
        If (k=0)
            Return poly ;
        Else
           Poly= a[k-1]+x*poly ;
           Return Eval( k-1 , a , x , poly) ;
    }
Appel initial Eval(n,a,x,a[n])
```

……..2pts

e) Arbre des appels…….2pts

Appel initial

↓ **poly=a[3]=4**

Eval(3, {4,10,0,5} , 2 , 4)

↓ **poly=a[2]+2*poly=18**

Eval(2 , {4,10,0,5} , 2 , 18)

↓ **poly=a[1]+2*18=36**

Eval(1 , {4,10,0,5},2,36)

↓ **poly=a[0]+2*36=7**

Eval(0 , {4,10,0,5} , 2 , 77)

↓ **poly =77**

CORRIGE-TYPE / EXAMEN OC1 / SESSION JANVIER 2017

Questions de cours (10 pts)

h) Le nombre d'appels est f(n) défini par f(1)=0 et f(n)=2*f(n/2)+1 d'où en majorant n par 2^n: f(n)≤ **n-1**…...1pt

i) Le nombre d'arrêtes est au max = C_n^2 = **n(n-1)/2** ……..…...1pt

j) Le nombre de solutions candidates pour le TSP symétrique est n!/(2*n) = **(n-1)!/2**…....1pt

k) Le nombre de solutions candidates pour le KSP simple = le nombre d'entiers formés de n bits = **2^n** ..…...1pt

l) $\begin{cases} f \in O(g) \\ g \in O(h) \end{cases} \begin{cases} \exists c \in R^+, \exists n_0 \in N, \forall x \geq n_0 : f(x) \leq c.g(x) \\ \exists c' \in R^+, \exists n_1 \in N, \forall x \geq n_1 : g(x) \leq c'.h(x) \end{cases}$ (par déf.)

$\begin{cases} \exists c \in R^+, \exists n_0 \in N, \forall x \geq n_0 : f(x) \leq c.g(x) \\ \exists c' \in R^+, \exists n_1 \in N, \forall x \geq n_1 : c.g(x) \leq c.c'.h(x) \end{cases}$

$\exists c \in R^+, \exists c' \in R^+, \exists n_2 \in N, \forall x \geq n_2 : f(x) \leq c.c'.h(x)$

(puisque ≤ est transitive en prenant n_2=max(n_0,n_1))

$\exists c'' \in R^+, \exists n_2 \in N, \forall x \geq n_2 : f(x) \leq c''.h(x)$ (en posant c.c'=c'')

$f \in O(h)$……...……2pt

m) La complexité de cet algo est défini par T(n)= 8T(n/2)+n^2 .

T(n) = 8[8T(n/4)+$(n/2)^2$]+n^2 (en substituant T(n/2) par 8T(n/4)+$(n/2)^2$).

= n^2+2n^2+8^2T($n/2^2$)

=…

= $2^0n^2 + 2^1n^2 + 2^2n^2 + 2^3n^2 + \ldots + 2^{\log n-1}n^2 + 8^{\log n}T(1)$ (en majorant n par 2^n).

= $(2^{\log n} -1)\, n^2 + 2^{3\log n} * T(1)$

= $(n -1)\, n^2 + n^3 * T(1)$

= **$O(n^3)$** (T(1) est bien évidemment constant)……..…..…...2pt

n) Le nombre d'itérations de la boucle interne est toujours pair et = i =2.k.

Le nombre d'itérations de la boucle externe est N/2 si N est pair et (N-1)/2 si N est impair

Le nombre d'affichages de * est alors $\begin{cases} \sum_{k=1}^{\frac{N}{2}} 2.k = 2.\frac{\left(\frac{N}{2}\right)\left(\frac{N}{2}+1\right)}{2} = \frac{N^2+2N}{4} \\ \sum_{k=1}^{\frac{N-1}{2}} 2.k = 2.\frac{\left(\frac{N-1}{2}\right)\left(\frac{N-1}{2}+1\right)}{2} = \frac{N^2-1}{4} \end{cases}$ ……..…..2pt

Exercice

1……3pt

Il faut passer le résultat M en paramètre pour éviter la remontée.

```
Int MaxTab(int tab[] , int k, int M ) ;
  {
     if (k = 1 )
        return tab[k];
     else
        {
          M = max(M , tab[k-1]) ;
          return MaxTab(tab, k-1, M ) ;
        }
  }
```

Appel initial MaxTab(tab, n , tab[n]) ;

Exercice 2

a) Approche gloutonne : on prend les objets selon l'ordre décroissant du rapport v_i/p_i :

Objet	1	2	3	4	5
v_i	16	19	23	28	25
p_i	2	1	4	4	1
v_i/p_i	8	19	5.75	7	25
Ordre	**3**	**2**	**5**	**4**	**1**
Etat du sac	2+2=4≤6	1+1=2≤6	/	4+4=8>6	0+1=1≤6
Variables x_i	1	1	0	0	1

Solution = {1,2,5} avec objectif = 16+19+25=60…….3.5pt

b) Programmation dynamique : on construit une table f(0..5, 0..6) (une matrice 6×7) telle que:
f(i,j) = valeur maximale des objets que l'on peut transporter si le poids maximal permis est j et que les objets que l'on peut inclure sont ceux numérotés de 1 à i.
La solution optimale sera alors en f(5 , 6) (la dernière case). Ainsi :
f(0,j) = 0 ∀j ; et pour i≥1 : f(i,j) = max(f [i-1, j), f [i-1, j-p_i) + v_i) ;

v_i	p_i	**f**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
		0	0	0	0	0	0	0	0
16	2	**1**	0	0	16	16	16	16	16
19	1	**2**	0	19	16	35	35	35	35
23	4	**3**	0	19	16	35	35	42	39
28	4	**4**	0	19	16	35	35	47	44
25	1	**5**	0	25	44	41	60	60	72

Objectif = 72 = <u>25</u>+47 = <u>25</u>+<u>28</u>+<u>19</u> = v_5+v_4+v_2 donc la solution est {2,4,5}……..3.5pt

Formation Master 2 IDO
Module Méthodes d'optimisation
Examen de rattrapage mars 2020

Exercice 1 (10 points)

1. **Citer deux avantages et deux inconvénients des métaheuristiques en général ?**
2. **Citer les paramètres d'intensification et de diversification pour chacune des métaheuristiques :**
   a) **Colonies de fourmis ;**
   b) **Algorithmes génétiques ;**
   c) **Colonies d'abeilles ;**
3. **Dans quels cas est préconisée une méthode approchée ?**
4. **Donner une grandeur pour mesurer l'approximation d'une solution.**
5. **Pourquoi utilise-t-on la notation préfixée pour représenter une solution en programmation génétique ?**

Exercice 2 (10 points)

Considérons un algorithme génétique appliqué à une population de 8 entiers (individus). Supposons que la fitness est $f(x) = x^3$ et qu'à la génération t la composition de la population est comme suit :

3	**3**	**5**	**5**	**5**	**5**	**7**	**7**	**7**

1) **Calculer les probabilités respectives de sélectionner chacun des individus 4 , 2 et 5 en utilisant la méthode de la roulette.**
2) **Calculer les probabilités respectives de sélectionner chacun des individus 4 , 2 et 5 en utilisant la méthode du rang.**
3) **Ecrire chacun des deux algorithmes de sélection par roulette et par rang.**
4) **Sachant que chaque individu est codé en binaire sur 4 bits, en appliquant un croisement en un point et une mutation qui change le dernier bit à droite, quels seront les individus pouvant constituer la population de la génération t+1 ?**

Formation Master 2 IDO
Module Méthodes d'optimisation
Examen de rattrapage session mars 2018

Exercice 1 (08 points)

On se propose de résoudre le problème KSP (problème du sac-à-dos simple) (n ; C ; v_i ; p_i ; i=1, n) à l'aide de la méthode PSO (optimisation par essaim de particules).

1. **Formulation mathématique :**
   a) **Donne une représentation d'une solution.**
   b) **Déduire la taille de l'espace de recherche.**
   c) **Ecrire la formulation mathématique du problème.**
2. **Application de PSO**
   a) **Ecrire l'algorithme d'une fonction qui teste si une solution est faisable.**
   b) **Définir les trois opérations nécessaires à la mise à jour d'une position d'une particule.**
   c) **Ecrire l'algorithme d'initialisation d'un essaim de k particules.**

Exercice 2 (06 points)

Expliquer les trois versions suivantes de l'algorithme ACO (optimisation par colonies de fourmis) appliqué au TSP (problème du voyageur de commerce) puis écrire l'algorithme de chacune d'elles :

a) **La version élitiste ;**
b) **La version min-max ;**
c) **La version 2-opt ;**

Exercice 3 (06 pts)

On se propose de trouver une régression pour la fonction réelle d'une variable réelle x définie par : $f(x)= x^2 - 3x + 2$ en utilisant la programmation génétique avec les paramètres suivants :

terminaux : (feuilles de l'arbre)	**x et une constante réelle (Cela signifie que les feuilles de l'expression LISP, seront soit x, soit un réel)**
Ensemble des fonctions : (nœuds de l'arbre)	**+, -, * (Cela signifie que les nœuds internes de l'arbre seront les opérations "classiques" +, - et ***
Ensemble choisi pour le calcul de la fitness :	**0 ; 1 ; -1 ; 2 ; 3**
Fitness :	**La somme, en valeur absolue, de la différence entre la valeur retournée par f(x) et la valeur de la S-expression pour les 5 valeurs.**

d) **Donner une population initiale de 4 individus de votre choix.**
e) **Ecrire ces quatre individus en lisp.**
f) **représenter ces individus en arbres.**
g) **Calculer la fitness de chacun des individus.**
h) **Décrire une opération de croisement de deux individus.**

Formation Master 1 IDO
OPTIMISATION COMBINATOIRE 2
Examen de rattrapage session juillet 2019

Questions de cours (06 pts)

1) **Expliciter les différences entre :**
   - **Méthode exacte et méthode approchée.**
   - **Méthode déterministe et méthode indéterministe.**
   - **Heuristique et métaheuristique**
2) **Quels sont les procédés que doit utiliser une méthode approchée pour garantir l'approximation de la solution fournie ?**
3) **Dans quels cas est préconisée une méthode approchée ?**
4) **Quel est l'impact d'une liste tabou trop large sur l'algorithme d'une recherche tabou ? Et quel en est l'impact si la liste est trop restreinte ?**

Exercice 1 (06 pts)

On se propose de fournir un algorithme de recuit simulé qui résout le problème du sac-à-dos simple en utilisant la fonction de voisinage qui associe à chaque solution x l'ensemble des solutions obtenues en remplaçant un objet de x par un autre.

1) **Donner une formulation mathématique du problème du sac-à-dos simple.**
2) **Ecrire l'algorithme qui retourne un voisin aléatoire d'une solution x.**
3) **Ecrire l'algorithme de recuit simulé résolvant ce problème.**

Exercice 2 (08 pts)

On se propose de fournir un algorithme de recherche locale qui calcule le minimum de la fonction réelle *f* à deux variables réelles x, y définie sur $[0;5]^2$ par : $f(x,y) = x^2-y^2+xy+1$.

1) **Qu'appelle-t-on ce type de problèmes d'optimisation ?**
2) **Donner une solution initiale (x_0,y_0) de votre choix à ce problème.**
3) **Considérons la fonction de voisinage V: $(x,y) \in [0;5]^2 \rightarrow [x-0,1 ; x+0,1] \times [y-0,1 ; y+0,1]$.**
   a. **Donner un voisin de votre choix à la solution initiale (x_0,y_0).**
4) **Exécuter les deux premières itérations d'une recherche locale.**
5) **Ecrire l'algorithme détaillé de recherche locale permettant de résoudre ce problème.**
6) **Modifier cet algorithme pour implémenter une descente aléatoire.**

Formation Master 2 OMID
Module Méthodes d'optimisation
Examen de rattrapage février 2019

Exercice 1 (07 points)

L'algorithme ci-desoous traite un problème d'optimisation à l'aide d'une métaheuristique qu'on a vue au cours.

```
float f(float x)
  return x*x+sin(x) ;
// Initialisation
Int n,m,maxiter ;
Scanf n ;
Scanf m ;
Scanf maxiter ;
Int i ;
Float x[] , v[] , pbest[] ;
For (i = 0; i<m; i++)
        {  x[i] = -3+6*rand(0,1) ;
           v[i] = 6*rand(0,1) ;
          pbest[i] = x[i] ; }
// main
Const infini= 1000000 ;
Float fopt = infini , gbest ;
For (iter = 0; iter < maxiter ; iter++)
   {  For (i = 0; i<m; i++)
         if (f(pbest[i]) < fopt)
              { gbest = pbest[i]  ; fopt= f(gbest) ; }
       For (i = 0; i<m; i++)
              { v[i] = c1* v[i] +c2*abs( x[i]-pbest[i])+c3*abs( x[i]- gbest) ;
                x[i] = x[i]+ v[i];    }
            if (f(x[i]) < f(pbest[i]))  pbest[i] = x[i] ; } }
cout gbest, fopt ;
```

1. **De quel problème s'agit-il ?**
2. **Qu'appelle-t-on cette méthode ?**
3. **Expliciter les paramètres de cette métaheuristique.**
4. **Calculer la complexité de cet algorithme.**

Exercice 2 (07 points)

Considérons un algorithme génétique appliqué à une population de 8 entiers (individus). Supposons que la fitness est $f(x) = x^2$ et qu'à la génération t la composition de la population est comme suit :

1	1	2	2	2	3	3	3

1. **Ecrire l'algorithme de sélection par roulette.**
2. **Calculer les probabilités respectives de sélectionner chacun des individus 1 , 2 et 3 en utilisant la méthode de la roulette.**
3. **Sachant que chaque individu est codé en binaire sur 4 bits, en appliquant un croisement en un point et une mutation qui change le dernier bit à droite, quels seront les individus pouvant constituer la population de la génération t+1 ?**

Exercice 3 (06 points)

Pour appliquer l'algorithme d'optimisation par colonie de fourmis pour résoudre un problème d'optimisation, deux étapes préliminaires sont nécessaires. Lesquelles ? expliquer ces deux étapes pour le problème su sac-à-dos simple (de capacité C, n objets ; de valeurs vi ; de poids pi ; i=1,n).

Formation Master 1 IDO
Module OPTIMISATION 2
Examen session rattrapage juin 2018

Exercice 1 (Questions de cours) (06 points)

1. **Pour résoudre un problème d'optimisation, dans quel cas a-t-on recours à une méthode approchée ?**
2. **Dans quel cas la méthode exacte est-elle préconisée ?**
3. **Citer deux critères d'arrêt du processus itératif qui pourraient être utilisés dans les métaheuristiques ?**
4. **Quelle action de l'algorithme du recuit simulé justifie-t-elle que ce dernier est indéterministe ?**
5. **Que signifie physiquement la fonction objectif dans l'algorithme du recuit simulé ?**
6. **Expliciter la diversification et l'intensification dans la recherche taboue.**

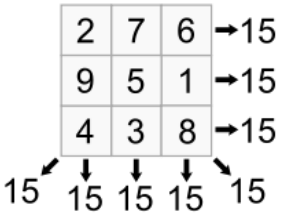
Exercice 2 (06 points)

On se propose d'appliquer une recherche locale aléatoire pour calculer le maximum de la fonction f à une variable réelle x définie par : $f(x) = -x^3 + 3x$ avec $x \in [-5, +5]$. Le voisinage d'une solution x est l'intervalle $[x - 0.1;\ x + 0.1]$.

1. **Ecrire l'algorithme implémentant cette méthode.**
2. **Calculer sa complexité.**

Exercice 3 (08 points)

Un carré magique d'ordre *n* est une matrice carrée n×n contenant les n^2 premiers entiers de 1 à n^2, de telle sorte que leurs sommes sur chaque ligne, sur chaque colonne et sur chaque diagonale principale soient égales. On appelle alors *constante magique* la valeur S(n) de ces sommes. On a prouvé que pour tout entier n > 2, il existe un carré magique d'ordre n et que $S(n) = (n+n^3)/2$.
Exemple : pour n=3 ; S(n)=15 (figure ci-contre).



Soit le problème : "trouver un carré magique d'ordre n". Sachant que ce problème est en général, NP-difficile, on se propose de le résoudre en utilisant la méthode de la descente.

1) **Proposer un codage pour une solution à ce problème.**
2) **Quelle est la taille de l'espace de recherche ?**
3) **Ecrire une procédure qui génère une solution aléatoire (approchée) à ce problème.**
4) **Ecrire les formules mathématiques que doit satisfaire une solution optimale.**
5) **Déduire une fonction fitness pour évaluer les solutions.**
6) **Proposer une fonction de voisinage pouvant être utilisée dans une recherche locale.**

Formation Master 2 OMID
Module Méthodes Emergentes
Examen de rattrapage mars 2017

Exercice 1 (Questions de cours) (06 points)

1. **Pour résoudre un problème d'optimisation, dans quels cas a-t-on recours à une méthode approchée ? 1.5**
2. **Citer quatre critères d'arrêt du processus itératif utilisés dans les métaheuristiques ? 2**
3. **Donner les paramètres qui justifient l'indéterminisme de chacune des métaheuristiques dans le tableau suivant : 2**

Métaheuristique	Paramètres d'indéterminisme
Algorithme génétique	
Colonie de fourmis	
Colonie d'abeilles	
Essaims de particules	

4. **Interpréter ce que signifie "la fonction objectif" d'un problème d'optimisation pour chacun des algorithmes : PSO, ACO, BCO. 1.5**

Exercice 2 (04 points)

L'algorithme ci-dessous permet de calculer une solution approchée du minimum d'une fonction réelle *f* à n variables entières :

```
Input : n, f (x1,x2,…,xn)
Parametres : int k, Max_iterations ;
             float a,b,c ;
             set function V;
Pour i = 1 à k
    si = un vecteur aléatoire de n entiers ; // solution candidate aléatoire
    vi = un vecteur de n entiers ;           // utilisés pour améliorer les si
    calculer f(si) ;                         // évaluer les solutions initiales
fin pour
Calculer globalbest ;                        // la meilleure solution
Pour chaque si
    calculer  localbesti (Vi) ;              // le meilleur voisin de si
pour iter = 1 à Max_iterations
        pour i=1 à k
            vi+1 = a*vi + b*(si - localbest) + c*(si - globalbest) ;
            si+1 = si + vi+1 ;
            mettre à jour globalbest et localbest;
        fin pour
fin pour
output globalbest , f(globalbest).
```

1. **Calculer la complexité de cet algorithme. 2**
2. **Cet algorithme implémente une métaheuristique qu'on a vue au cours, laquelle ? 0.5 Expliciter ses paramètres. 1.5**

Exercice 1 (10 pts)

On se propose de calculer les entiers a,b,c,d compris entre 1 et 20 qui satisfont 2a+b+3c+4d=50 en utilisant un algorithme génétique. Une solution peut être codée par un quadruplet (a,b,c,d) de $[1,20]^4$.

1. **Calculer la taille de l'espace de recherche ? 0.5**
2. **Proposer une fonction fitness. 0.5**
3. **Le tableau ci-contre fournit une population initiale de 5 individus. Evaluer les individus de cette population. 2.5**
4. **Appliquer une sélection par rang à la population initiale. 1**
5. **Ecrire l'algorithme de croisement en un seul point. 1.5**
6. **On considère un opérateur de mutation qui consiste à remplacer un gène du chromosome par un nouveau gène aléatoire. Ecrire l'algorithme correspondant. 1.5**
7. **Calculer la population de la deuxième génération en appliquant ces trois opérateurs.**

1	**5**	**2**	**10**
2	**3**	**5**	**4**
1	**8**	**2**	**4**
6	**10**	**1**	**7**
8	**2**	**9**	**5**

Formation Master 2 IDO
Module Méthodes d'optimisation
Examen de remplacement février 2020

Exercice 1 (10 points)

1. **Dans quels cas est préconisée une méthode approchée ?**
2. **Donner une grandeur pour mesurer l'approximation d'une solution.**
3. **Pourquoi utilise-t-on la notation préfixée pour représenter une solution en programmation génétique ?**
4. **Citer deux avantages et deux inconvénients des métaheuristiques en général ?**
5. **Citer les paramètres d'intensification et de diversification pour chacune des métaheuristiques :**

a) **Colonies de fourmis ;**
b) **Algorithmes génétiques ;**
c) **Colonies d'abeilles ;**

Exercice 2 (10 points)

Considérons un algorithme génétique appliqué à une population de 8 entiers (individus). Supposons que la fitness est $f(x) = x^2$ et qu'à la génération t la composition de la population est comme suit :

2	**2**	**2**	**2**	**5**	**4**	**4**	**4**

a) **Calculer les probabilités respectives de sélectionner chacun des individus 4 , 2 et 5 en utilisant la méthode de la roulette.**
b) **Calculer les probabilités respectives de sélectionner chacun des individus 4 , 2 et 5 en utilisant la méthode du rang.**
c) **Ecrire chacun des deux algorithmes de sélection par roulette et par rang.**
d) **Sachant que chaque individu est codé en binaire sur 4 bits, en appliquant un croisement en un point et une mutation qui change le dernier bit à droite, quels seront les individus pouvant constituer la population de la génération t+1 ?**

Formation Master 1 IDO
OPTIMISATION 2
Examen de remplacement mai 2017

Questions de cours (07 pts)

a) **Donner un exemple d'une métaheuristique à solution unique et un exemple d'une métaheuristique à population de solutions.**
b) **Donner un exemple d'une métaheuristique transformative et un exemple d'une métaheuristique constructive.**
c) **Citer un avantage et un inconvénient de la méthode de la descente.**
d) **Dans quels cas est préconisée une méthode exacte ?**
e) **Dans une recherche locale, quel est l'impact d'un voisinage trop large sur l'algorithme ? Et quel en est l'impact d'un voisinage trop restreint ?**
f) **Donner un outil pour mesurer l'approximation d'une solution approchée.**
g) **Que devient le recuit simulé lorsque la température est élevée ? Et qu'en est-il pour les basses valeurs de la température ?**

Exercice 1 (06 pts)

En appliquant la méthode du recuit simulé pour un certain problème, on est arrivé à une itération où t=5 et la valeur de la fonction objectif de la solution courante est 15. Cette solution a 3 voisins dont la valeur de la fonction objective donne 16, 18 et 20 respectivement. Pour chacune de ces solutions, on se propose de déterminer la probabilité qu'elle soit choisie pour être la prochaine solution courante.

a) **Déterminer ces probabilités pour un problème de maximisation.**
b) **Même question pour un problème de minimisation.**

Exercice 2 (07 pts)

On se propose de fournir un algorithme de recuit simulé qui résout le problème du sac-à-dos simple en utilisant la fonction de voisinage qui associe à chaque solution x l'ensemble des solutions obtenues en remplaçant un objet de x par un autre.

a) **Donner une formulation mathématique du problème du sac-à-dos simple.**
b) **Ecrire l'algorithme qui retourne un voisin aléatoire d'une solution x.**
c) **Ecrire l'algorithme de recuit simulé résolvant ce problème.**

Formation master 2 IDO S1 2018/2019

Module MEO

INTERROGATION ECRITE **2**

Pour appliquer l'algorithme PSO au problème du sac-à-dos simple (n ; v_i ; p_i ; i = 1,n) :

1) Implémenter les termes suivants :
   a. Une position d'une particule ;
   b. Une vélocité ;
   c. Une différence de positions ;
   d. Une somme de vélocités ;
   e. Le produit d'une vélocité par un scalaire ;
   f. Une somme d'une position et d'une vélocité

2) Appliquer les deux formules de mise à jour de la position et de la vélocité à un exemple en prenant c1=0.5 et c2=c3=2.

Formation master 2 IDO S1 2018/2019

Module MEO

INTERROGATION ECRITE

1) Dans quels cas est préconisée une méthode approchée ?
2) Quelle est la différence primordiale entre AG et PG ?
3) Citer deux avantages et deux inconvénients des métaheuristiques en général ?
4) Citer les paramètres d'intensification et de diversification pour chacune des métaheuristiques :
   a) Colonies de fourmis ;
   b) Algorithmes génétiques ;
   c) Colonies d'abeilles ;

Formation master 1 IDO S2 2017/2018

Module Opti. 2.

INTERROGATION ECRITE

Vous disposez de 10 cartes numérotées de 1 à 10. Vous devez choisir une façon de diviser celles-ci en 2 piles P_1 et P_2 de telle sorte que la somme des numéros des cartes de la première pile soit aussi proche que possible de 36. Chaque carte pouvant être soit dans P_1, soit dans P_2. On désire trouver la meilleure répartition des cartes.

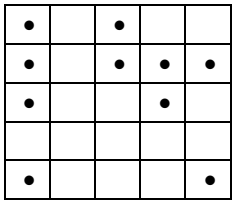
1) Trouvez un encodage pour l'ensemble de ces solutions.
2) Déduire la taille de l'espace de recherche.
3) Trouvez une fonction fitness permettant d'évaluer la qualité d'une solution.
4) Proposez une fonction de voisinage pour résoudre ce problème à l'aide d'une recherche locale..

Formation master 2 IDO S1 2017/2018

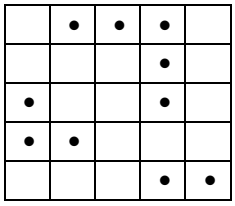
Module MEO

INTERROGATION ECRITE

Sur une grille à n lignes et n colonnes, on désire placer k ($k < n^2$) jetons de telle sorte que le nombre de jetons placés sur chaque ligne et le nombre de jetons placés sur chaque colonnes soient pairs. Exemple : n =5 , k = 10 :



Solution faisable.



Solution non faisable

1. Calculer la taille de l'espace de recherche.
2. On désire résoudre ce problème par un AG.

a) Proposer un codage aux solutions du problème.

b) Proposer une fonction fitness.

3. Si l'on veut résoudre ce problème par un algorithme de colonie de fourmis :

c) Montrer comment une fourmi devrait-elle construire une solution.

d) Définir une distance entre les composantes d'une solution.

MASTER 1 IDO
OPTIMISATION COMBINATOIRE 2 / S2 2019-2020 /
TD0 / RAPPELS

Exercice 1 Triangle de Pascal

On veut calculer les coefficients binomiaux

$$C_n^k = \left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}.$$

Rappelons les propriétés suivantes :

- $\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) + \left(\begin{array}{c} n-1 \\ k \end{array} \right)$ pour $0 < k < n$,
- $\left(\begin{array}{c} n \\ n \end{array} \right) = 1$ et $\left(\begin{array}{c} n \\ 0 \end{array} \right) = 1.$

Question 1.1 Donner un algorithme récursif du calcul de $\left(\begin{array}{c} n \\ k \end{array} \right)$. Evaluer sa complexité.

Question 1.2 Ecrire l'algorithme qui retourne $\left(\begin{array}{c} n \\ k \end{array} \right)$ en utilisant la technique de la programmation dynamique. Evaluer sa complexité.

Exercice 2 Problème du stockage

Considérons n programmes $P_1,P_2,...,P_n$ qui peuvent être stocker sur un disque dur de capacité D gigabytes.

— Chaque programme P_i a besoin s_i gigabytes pour être stocké et a une valeur v_i

— Tous les programmes ne peuvent pas être stockés sur le disque : $\sum_{i=1}^{n} s_i > D.$

Les programmes stockés dans le disque dur doivent maximiser la valeur totale, sans dépasser la capacité du disque dur. L'objectif est de concevoir un algorithme qui permet de calculer un tel ensemble.
Nous allons construire un tableau T dans lequel les lignes seront indexées par les programmes et les colonnes par les valeurs. L'élément $T[i,j]$ représentera la valeur maximale pour un disque dur de capacité j à l'aide des i premiers programmes.

Question 2.1 Donner la formule de récurrence.

Question 2.2 Donner l'algorithme utilisant la programmation dynamique.

Question 2.3 Donner la complexité de cet algorithme.

Exercice 3 Problème Le chemin le plus long dans un graphe

Soit $G = (V,E)$ un graphe orienté avec $V = \{v_1,...,v_n\}$. On dit que G est ordonné s'il vérifie les deux propriétés suivantes :

1. Chaque arc de ce graphe est de la forme $(i \rightarrow j)$ si $i < j$
2. Tous les sommets sauf le sommet v_n ont au moins un arc sortant.

Ici, par souci de simplification, nous supposerons qu'il existe un chemin allant de v_i vers v_n pour tout $i = 1,...,n$.

L'objectif est de trouver le chemin le plus long entre les sommets v_1 et v_n.

Question 3.1 Montrer que l'algorithme glouton suivant ne résout pas correctement le problème.

1. $u \leftarrow v_1$;

2. $L \leftarrow 0$;
3. Tant qu'il existe un arc sortant du sommet u
   (a) choisir l'arc ($u \rightarrow v_j$) tel que j est le plus petit possible
   (b) $u \leftarrow v_j$;
   (c) $L \leftarrow L + 1$;
4. retourner L

Question 3.2 Donner la formule de récurrence qui permet de calculer la longueur du chemin le plus long commençant par v_1 finissant par v`.

Question 3.3 Donner un algorithme qui retourne la longueur du chemin le plus long commençant par v_1 finissant par v_n.

Question 3.4 Modifier l'algorithme précédent afin qu'il retourne le chemin.

Question 3.5 Donner la formule de récurrence permettant de calculer le chemin de poids maximum commençant par v_1 finissant par v_n.

Question 3.6 En d´déduire l'algorithme.

Question 3.7 Donner la formule de récurrence qui permet de calculer le chemin de poids maximal de ` arcs commençant par v_1 finissant par v_i . En déduire l'algorithme.

Exercice 4 Planning

Considérons un chef de projet qui doit gérer une équipe en lui affectant un projet qui dure une semaine. Le chef de projet doit choisir si il prend un projet stressant ou non stressant pour la semaine.

1. Si le chef de projet choisit le projet qui n'est pas stressant durant la semaine i, alors l'entreprise reçoit un revenu j.
2. Si le chef de projet choisit le projet qui est stressant durant la semaine i, alors l'entreprise recroit un revenu h_i et l'équipe ne travaille pas durant la semaine $i - 1$

Exemple : Si chef de projet choisit de se reposer à la semaine 1, puis de prendre le projet stressant à la semaine 2, puis de prendre les projets non-stressant à la semaine 3 et 4. Le revenu total est de 70 et il correspond au maximum.

semaine 1	semaine 2	semaine 3	semaine 4
10	1	10	10 h 5 50
5	1		

Question 4.1 Montrer que l'algorithme suivant ne résout pas correctement le problème.

1. pour chaque itération $i = 1,...,n$
   (a) si $h_{i+1} >$ `$_i$ + `$_{i+1}$ alors
      i. Choisir ne pas travailler à la semaine i
      ii. Choisir le projet stressant à la semaine i+1
      iii. Continuer à literation $i + 2$
   (b) sinon
      i. Choisir le projet non-stressant à la semaine i
      ii. Continuer à literation $i + 1$

Question 4.2 Donner un algorithme qui retourne le revenu maximal que peut obtenir le chef de projet.

Exercice 5 Multiplications chainées de matrices.

On veut calculer le produit de matrices $M = M_1M_2 \cdots M_n$. Multiplier une matrice $p \times q$, par une matrice $q \times r$ en utilisant la méthode standard nécessite pqr produits scalaires.

Question 5.1 Considérons 4 matrices A : 20×5, B : 5×100, C : 100×8, D : 5×30. On veut calculer le produit $ABCD$. En fonction des parenthétisations, le nombre de produits varie.
D´déterminer le nombre de produits pour calculer $ABCD$, si on utilise les parenthétisations suivantes : $((AB)C)D$ ou $(A(BC))D$
Question 5.2 Ecrire une formule de récurrence pour calculer $c(i,j)$.
Question 5.3 Ecrire un algorithme utilisant la programmation dynamique

Exercice 6.

1. Donner un algorithme de programmation dynamique pour résoudre le problème suivant :
   Entrée : une matrice A de taille $n \times m$ où les coefficients valent 0 ou 1.
   Sortie : la largeur maximum K d'un carré de 1 dans A, ainsi que les coordonnées (I, J) du coin en haut à gauche d'un tel carré (autrement dit pour tout i, j, $I \le i \le I + K - 1$, $J \le j \le J + K - 1$, $A[i, j] = 1$).
2. Quelle est sa complexité ?

Exercice 7.
La bibliothèque planifie son déménagement. Elle comprend une collection de n livres $b_1, b_2, \ldots, b_n$. Le livre b_i est de largeur w_i et de hauteur h_i. Les livres doivent être rangés dans l'ordre donné (par valeur de i croissante) sur des étagères identiques de largeur L.

1. On suppose que tous les livres ont la même hauteur $h = h_i, 1 \le i \le n$. Montrer que l'algorithme glouton qui range les livres côte à côte tant que c'est possible minimise le nombre d'étagères utilisées.
2. Maintenant les livres ont des hauteurs différentes, mais la hauteur entre les étagères peut se régler. Le critère à minimiser est alors l'encombrement, défini comme la somme des hauteurs du plus grand livre de chaque étagère utilisée.
   (a) Donner un exemple où l'algorithme glouton précédent n'est pas optimal.
   (b) Proposer un algorithme optimal pour résoudre le problème, et donner son coût.
3. On revient au cas où tous les livres ont la même hauteur $h = h_i, 1 \le i \le n$. On veut désormais ranger les n livres sur k étagères de même longueur L à minimiser, où k est un paramètre du problème. Il s'agit donc de partitionner les n livres en k tranches, de telle sorte que la largeur de la plus large des k tranches soit la plus petite possible.
   (a) Proposer un algorithme pour résoudre le problème, et donner son coût en fonction de n et k.
   (b) On suppose maintenant que la taille d'un livre est en $2^{o(kn)}$. Trouver un algorithme plus rapide que le précédent pour répondre à la même question.

Formation Master 1 IDO
Optimisation combinatoire 2
Année universitaire 2019/2020

TD1 : Généralités

Exo1

Donner la taille de l'espace de recherche de chacun des problèmes suivants puis proposer une heuristique et un algorithme glouton de résolution :

1) Problème d'ordonnancement de n taches indépendantes sur une seule machine ;
2) Problème d'ordonnancement de n taches indépendantes sur deux machines ;
3) Problème de 8 reines sur un échiquier ;
4) Problème du loup, la chèvre et du chou ;
5) Problème de la chaine eulérienne ;
6) Problème de la chaine ha miltonienne.

Exo2

L'heuristique du plus proche voisin pour le PVC s'énonce comme suit :
On choisit un sommet (ville) arbitraire.
On part au sommet voisin le plus proche, puis de celui-là, à son plus proche voisin non visité, etc...;
Jusqu'à ce que tous les sommets aient été parcourus, où l'on revient au départ.

1) Donner un modèle puis une formulation mathématique pour le PVC.
2) Rappeler la taille de l'espace de recherche.
3) Ecrire l'algorithme de recherche exhaustive.
4) Ecrire l'algorithme de l'heuristique ci-dessus.
5) Exprimer sa complexité.

Exo 3

On veut colorer un graphe avec le nombre minimum K de couleurs.

1. Est-ce un problème d'optimisation ?
2. De quel type ?
3. Quelle est la forme des solutions ?
4. Que peut-on dire de l'espace de recherche ?

Exo4

Considérons les trois heuristiques suivantes pour le KSP (C ; n ; vi ; pi ; i=1,n).
H1: on commence par l'objet de plus petit poids p_i.
H2: on commence par l'objet de plus grande valeur v_i.
H3: on commence par l'objet ayant le plus grand rapport v_i/p_i.

1) Ecrire l'algorithme de chacune de ces heuristiques puis évaluer leurs complexités.

2) Comparer ces heuristiques a une méthode exhaustive puis a une méthode exacte telle que la PD.
3) Proposer votre propre heuristique puis comparer tous les résultats dans un tableau récapitulatif.
4) Tracer les courbes des temps d'exécutions et celles des résultats.
5) Que peut-on conclure ?

Exo 5

Un sodoku 4x4 se compose d'une grille de 4x4 cases ; divisée en 4 régions de 2x2 cases; la grille est déjà partiellement remplie ; il faut la compléter avec des nombres entre 1 et 4 ; de telle sorte qu'un chiffre n'apparaisse jamais 2 fois dans chaque ligne, chaque colonne et chaque région. Caractériser puis formuler ce problème d'optimisation.

Formation Master 1 IDO
Optimisation combinatoire 2
Année universitaire 2019/2020

TD2 : RECHERCHES LOCALES

Exo1

La variable du problème 3-SAT est un vecteur binaire de n bits. Considérons une fonction de voisinage qui associe à chaque solution x l'ensemble des solutions dont la distance de Hemming à x est 1.

1) Donner la taille de l'espace de recherche.
2) Donner le voisinage de la solution (1,0,0,1,1,0) pour n = 6.
3) Ecrire l'algorithme calculant le voisinage d'une solution x.
4) Ecrire l'algorithme d'une recherche aléatoire utilisant ce voisinage pour ce problème.

Exo2

Considérons, pour le PVC symétrique de n villes, une fonction de voisinage qui associe à chaque solution x l'ensemble des solutions obtenues par permutations de deux villes quelconques.

1) Donner la taille de l'espace de recherche.
2) Donner le voisinage de la solution (1,2,3,4,5,6) pour n = 6.
3) Ecrire l'algorithme calculant le voisinage d'une solution x.
4) Ecrire l'algorithme d'une recherche de recuit simulé pour ce problème.

Exo3

Soit le problème de programmation non convexe : Max $f(x)=x^3 - 60x^2 + 900x + 100$

$0<=x<=31$

(a) Utiliser les dérivées premières et secondes pour déterminer les points critiques (avec les solutions limites de la région réalisable) où x est un maximum ou un minimum local.
(b) Dessiner la fonction f.
(c) En utilisant x=15,5 comme solution initiale, faire la première itération du recuit simulé.
(d) En utilisant x=15,5 comme solution initiale, résoudre à l'aide de IOR Tutorial; relever, pour chaque itération, le nombre de candidats rejetés avant qu'il y en ait un sélectionné, ainsi que le nombre d'itérations où une solution qui n'améliore pas l'objectif est sélectionnée.

Exo4

En appliquant la méthode de recuit simulé pour certain problème, vous êtes arrivé à une itération où t = 2 et la valeur de la fonction objectif de la solution courante est 30. Cette solution a 4 voisins dont la valeur de la fonction objective donne 29, 34, 31 et 24. Pour chacune de ces solutions vous voulez déterminer la probabilité qu'elle soit choisie pour être la prochaine solution courante.
(a) Déterminer ces probabilités pour un problème de maximisation.
(b) Même question pour un problème de minimisation.

Exo5

Une entreprise dispose de plusieurs dépôts (Di) contenant chacun un certain nombre de containers. Différents magasins (Mj) commandent des containers. On connaît le coût de transport de chaque dépôt aux magasins.
Exemple :

	M1	M2	M3	DEPOTS
D1	5	3	4	8
D2	6	7	2	9

Les demandes magasins sont 4, 5 et 8 containers.
Quelle est l'organisation des livraisons des containers pour minimiser le coût total de transport ?
Ce problème peut se modéliser par programmation linéaire en nombres entiers et sa résolution peut se faire par séparation et évaluation (Chercher une solution).
Proposez une solution pour le problème en se basant sur la méthode de recherche taboue.

Exo6

Soit le problème de coloration des arrêtes d'un graphe G(X, E) avec X l'ensemble des sommets et E l'ensemble des arrêtes. L'objectif est de colorer les arrêtes du graphe tel que deux arêtes ayant une extrémité commune soient de couleurs différentes. Soit ci la couleur de l'arrête i. Le but de l'exercice est d'appliquer la méthode de recherche taboue pour le problème.
Donnez la représentation d'une solution du problème, montrer à l'aide d'un graphe cette représentation.
Considérons trois couleurs c1, c2 et c3 et selon votre représentation, donnez une solution s initiale.
Puis déroulez l'algorithme tabou pour deux itérations.
Pour les deux itérations, donnez-le contenu de la liste taboue.

Exo 7

1) Ecrire un algorithme d'affectation de fréquences utilisant la méthode du recuit simulé.
2) Pour cela définir la fonction énergie qui doit être minimisée.
3) Définir une transformation élémentaire dans l'espace des configurations.
4) Indiquer dans quelle(s) condition(s) une transformation élémentaire peut être acceptée.
5) Donner une méthode permettant de construire la configuration initiale sur laquelle le recuit simulé est appliqué.
6) A quel moment en théorie, dans le déroulement de l'algorithme, la température devrait-elle être réduite ?

Bibliographie

G. Colson, Chr. De Bruyn. Models and methods in multiple criteria decision making, Pergamon, Oxford, 1989.
K. Miettinen. On the methodology of multiobjective optimization with applications. Report 60, University of Jyvaskyla, Departement of Mathematics, Jvvaskyla, 1994.
R.L. Keeney, H. Raiffa. Decision with multiple objectives: preferences and values trade-offs. Wiley, 1976.
L.Y. Maystre, J. Pictet, J. Simos. Méthodes multicritères ELECTRE. Presses polytechniques et universitaires romandes, 1994.
B. Roy, D. Bouyssou. Aide multicritère à la décision : méthodes et cas", Economica, 1993.

J.C. Pomerol and S. Barba-Romero. Multicriterion decision in management: principles and practice,
Kluwer Academic Publishers, 2000.
P. Vallin , D. Vanderpooten. Aide à la decision. Une approche par les cas. Ed. Ellipses, Paris, 2002.

Formation Master 1 IDO
Optimisation combinatoire 2
Année universitaire 2017/2018

TP1 COMPARAISON D'HEURISTIQUES

Considérons les approches suivantes pour le KSP de l'exo 4 TD1 (C ; n ; vi ; pi ; i=1,n).
FB : algorithme de force brute (recherche exhaustive) ;
DP : algorithme de programmation dynamique ;
H1 : on commence par l'objet de plus petit poids p_i ;
H2 : on commence par l'objet de plus grande valeur v_i ;
H3 : on commence par l'objet ayant le plus grand rapport v_i/p_i ;

6) Ecrire l'algorithme de chacune de ces approches puis évaluer leurs complexités.
7) Construire les tableaux des moyennes des résultats et des temps CPU. (on prend les tailles n=10,20,50,100,1000,10000,100000 et h=0.2,0.4,0.6,0.8 à raison de 5 instances pour chaque cas).
8) Tracer les courbes représentatives.
9) Que peut-on conclure ?
10) Modifier les intervalles des instances en utilisant un coefficient r=10,100,10000,1000000.
11) Refaire les courbes des temps CPU.
12) Que peut-on conclure ?

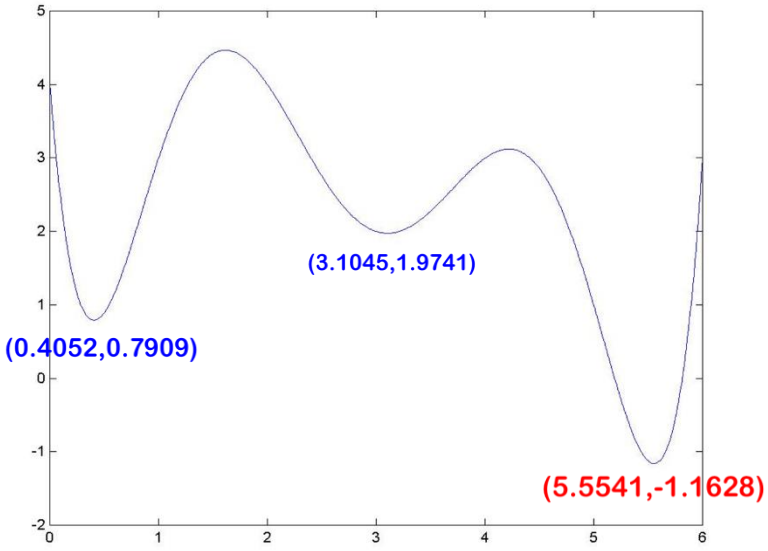
MASTER 1 IDO
OPTIMISATION COMBINATOIRE 2 / S2 2019-2020 /
TP2 / RECHERCHES LOCALES

Dans ce TP, on se propose de rechercher le minimum de la fonction réelle f d'une variable réelle x définie par :

$F(x) = 4 - 19.0167x + 36.39167x^2 - 25.2917x^3 + 8.041667x^4 - 1.19167x^5 + 0.0666676x^6$

(Ce type de fonction est souvent rencontrée dans des calculs scientifiques tels qu'en chimie, en génie civil, en mécanique, etc,…)

La figure ci-dessous illustre la représentation graphique de cette fonction :



1. **Quel est le type de ce problème d'optimisation ?**
2. **Que représentent les coordonnées indiquées sur cette figure ?**
3. **Expliciter l'espace de recherche de ce problème.**
4. **Justifier le recours à une méthode approchée.**
5. **Implémenter quatre métaheuristiques pour résoudre ce problème :**
   LS : Local Search
   HL : Hill Climbing
   SA : Simulated Annealing
   TS : Tabu Search
6. **Exécuter chaque métaheuristique 100 fois et récapituler les résultats dans un tableau puis tracer l'histogramme correspondant. Discuter puis interpréter ces résultats.**
7. **Implémenter les deux variantes de HL : First Deep et Best Deep puis comparer entre les trois approches.**
8. **Ajuster le paramètre T du SA en prenant en considération la qualité de la solution et le temps de calcul.**

Master 2 IDO 2019/2020

Optimization methods

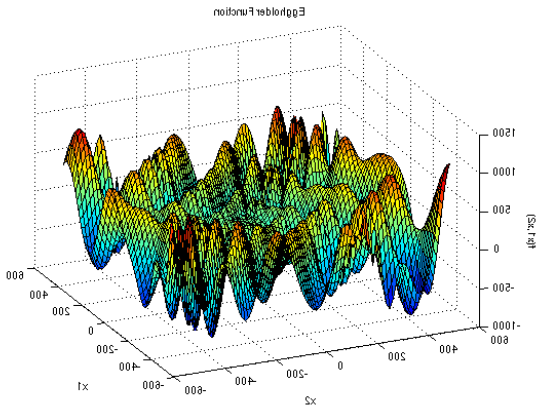
Practice work 1 / genetic algorithms

Purpose

This practice work consists to apply GA for optimizing one the famous test functions for optimization, which known with a great number of local minima that is called Eggholder function defined as ;

$$f(\mathbf{x}) = -(x_2 + 47)\sin\left(\sqrt{\left|x_2 + \frac{x_1}{2} + 47\right|}\right) - x_1 \sin\left(\sqrt{|x_1 - (x_2 + 47)|}\right)$$

This function is usually evaluated on the square $x_i \in [-512, 512]$, for all i = 1, 2.



$$f(\mathbf{x}^*) = -959.6407, \text{ at } \mathbf{x}^* = (512, 404.2319)$$

Goals:

- Understanding optimization concepts ;
- Implementing basic GA ;
- Adjusting parameters to improve GA efficiency;
- Discovering GA power ;
- Comparison between binary and decimal encoding ;
- Comparison between different genetic operator types.

Part 1

- Write the mathematical formulation for this problem.
- What is type of this optimization problem?
- Determine the search space.

Part 2

- Implement a basic GA for solving this problem using decimal encoding.
  (take NumIter = 200 ; m = 50 ; pc = 0.7 ; pm = 0.05 ; roulette wheel selection ; one-point crossover ; bit flip mutation).
- Adjusting parameters :
  - ✓ Number of iterations: construct the curve that represent the deviation ratio of means of 10 runs in terms of NumIter). Choose ideal value of NumIter.
  - ✓ Perform the same process for population size m ;
  - ✓ Rake pm=0 then pm=1 and run your program. Write then interpret your results.
  - ✓ Perform the same process for pc.

Master 2 IDO 2019/2020
Optimization methods
Tutorials 1 / genetic algorithms

Exercise 1

Consider a population of simple creatures, with a single chromosome of length n = 1000. Each entry in the chromosome can take four values (A, C, G, T). Assume that the population size is equal to M.

1. How many possible chromosomes are there?
2. Assuming that the chromosome length and the population size remain constant, what is the upper limit of the number of different chromosomes evaluated in the course of G generations?
3. If the population size is constant and equal to 10^{12}, how large a fraction q of the total number of chromosomes will be evaluated during 10^9 generations, assuming that all evaluated chromosomes are different?

Exercise 2

Consider the simple genetic algorithm applied on a population of 8 integer numbers. Suppose that at time t of the evolution the population has the following composition:

- $x=1$: 2 copies
- $x=2$: 3 copies
- $x=3$: 3 copies

Assuming that the fitness function is $f(x)=x^2$, calculate the probability of selecting the individuals $x=1$, $x=2$, and $x=3$ using roulette wheel selection.

Exercise 3

Consider a population consisting of five individuals with the fitness values (before ranking) $f_1 = 5, f_2 = 7, f_3 = 8, f_4 = 10, f_5 = 15$. Compute the probability that individual 4 will be selected (in a single selection step) with

(a) roulette wheel selection
(b) tournament selection, with tournament size equal to 2, and the probability of selecting the best individual (in a given tournament) equal to 0.75
(c) roulette wheel selection, based on linearly ranked fitness values, where the lowest fitness value is set to 1 and the highest fitness value set to 10.

Exercise 4

Calculate the probability that a binary chromosome of length L will not be changed by applying the usual bit-flip mutation with probability $p_m = 1/L$

Exercise 5

Write an evolutionary algorithm that searches for the shortest route between *N* cities. Use an encoding method such that the chromosomes consist of lists of integers determining the indices of the cities. Examples of five-city paths starting in city 4 are e.g. (4,3,1,2,5),(4,1,5,2,3),(4,5,1,2,3) etc. The first chromosome thus encodes the path 4 → 3 → 1 → 2 → 5 → 4. The fitness should be taken as the inverse of the route length (calculated using the ordinary cartesian distance measure, i.e. not the city-block distance measure). The program should always generate syntactically correct routes, i.e. routes in which each city is visited once and only once until, in the final step, the tour ends by a return to the starting city. Specialized operators for crossover and mutation are needed in order to ensure that the paths are syntactically correct.

(a) Define a mutation operator for the TSP that maps valid chromosomes (i.e. paths) onto other valid chromosomes.

(b) Define a crossover operator for the TSP that maps valid chromosomes onto other valid chromosomes.

(c) Using the specialized crossover and mutation operators, write an Evolutionary Algorithm that solves the Travelling Salesman Problem.

Exercise 6

Implement an Evolutionary Algorithm that solves the 8-queens problem.

Exercise 7

Consider a generational GA that has a population size of 100 individuals and uses roulette–wheel selection. Suppose that after running for *t* generations, the mean population fitness is 76.0 and that in the population there is only one copy of the best member, which has fitness 157.0. Also suppose that parents parent selection is performed on the population

- What is the expected number of copies of the best individual in the set of selected parents?
- What it the probability that there will be no copies of the best individual in the selected parents?

Exercise 8

Consider a population containing four individuals with chromosomes 101010, 000111 , 010101, and 011011, and fitness values 1,2,3 and 4 respectively. In a given selection step, assume that individual 1 (with chromosome 101010) has been selected (using roulettewheel selection) as the first parent. What is the probability that the schema 10xxxx will be represented in either of the two individuals resulting from the selection of a second parent, followed by crossover? (Crossover may occur, with equal probability, at any of the five available crossover points).

Exercise 9

Write a GA to find four integer numbers a,b,c,d such as : a +2 b +3 c +4 d = 30.
Run manually two iterations of this GA using parameters value of your choice.

Exercise 10

We want to look for the integer number of [0, 15] that maximize the function $f(x) = \frac{1}{4}\left|15x^2 - x^3\right| + 4$

Solve this problem using a GA with binary encoding and a population of 6 individuals.

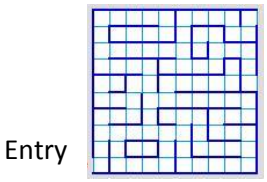
Exercise 11

You have 10 cards numbered 1 to 10. You have to divide them into two piles so that: the sum of the first pile is as close as possible to 36 and the product of all in the second pile is as close as possible to 360.

1. Determine individuals encoding for this problem.
2. Define a fitness function to evaluate solutions.

Exercise 12

We propose to find the path to the exit of a labyrinth.



Labyrinth

1. Determine individuals encoding for this problem.
2. Define a fitness function to evaluate solutions.

Exercise 13

We wish to realize a 2D tower with N bricks of width: 2,3,5 and 7 units. We want to build the tower, which is at the same time the highest, and the most solid. To evaluate this last criterion, we have a simulator capable of applying to the tower the combined effect of gravity and jolts of a given amplitude. After the calculation, the simulator returns a tower, potentially different, resulting from the possible displacement of the bricks during the simulation as well as the number of bricks having moved. The simulator also allows knowing the height of the tower (height of the highest element).
1. Determine a fitness function and the associated assessment procedure.
2. It is assumed that a tower cannot be more than 32 units wide. We propose you the following coding:
- A solution is a sequence of N 7-bit words.

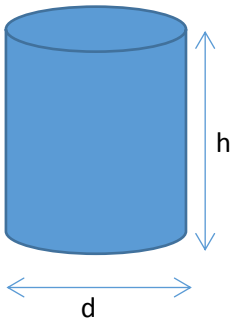
- A word represents the type of brick (coded on 2 bits) and a horizontal position (on 5 bits) to which one deposits a brick. The brick is then deposited as low as possible on the chosen column. Example: 01 00100 corresponds to a brick of size 3 arranged at 4 units of the extreme left point. What are the qualities and defects of this coding?

3. Propose another way to represent a tower by seeking to remove the flaws you identified in the previous question.

Exercise 14

Solve the following problem using a GA:

$$\begin{aligned} \text{Minimize} \quad & f(d,h) = c((\pi d^2/2) + \pi dh), \\ \text{Subject to} \quad & g_1(d,h) \equiv (\pi d^2 h/4) \geq 300, \\ \text{Variable bounds} \quad & d_{\min} \leq d \leq d_{\max}, \\ & h_{\min} \leq h \leq h_{\max}. \end{aligned}$$



Take c= 0.654 ; $d_{min} = h_{min} = 0$; $d_{max} = h_{max} = 20$;

Exercise 15

Maxf(x)=sin(x)

$0 \leq x \leq \pi$

Consider 6 bit string to represent the solution, then 000000 = 0 and 111111 = π.

Assume population size of 4.

Exercise 16

$$Minimize\ f = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$$

$0 \leq x \leq 10$; $0 \leq y \leq 10$

Références bibliographiques :

1. J. Hertz, A. Krogh & R. G. Palmer. An introduction to the theory of Neural Computation. (Addison-Wesley)
2. D. Michie, D.J. Spiegelhalter & C.C. Taylor. Machine Learning, Neural and Statistical Classification. (Ellis Horwood)
3. P. Naïm, P.H. Wuillemin, Ph. Leray, O.Pourret, A. Becker. Réseaux Bayésiens (Eyrolles)
4. http://www.librecours.org/cgi-bin/domain?callback=info&elt=190
5. http://asi.insa-rouen.fr/enseignement/siteUV/rna/

Master 2 IDO 2019/2020

Optimization methods

Recitation 2 / genetic programming

Exercise 1

Convert in prefix form:

1) f(x)= 3x+2
2) f(x)= $2x^3$-x+3
3) f(x)= x^2 +1 si x ≥ 0 et f(x)=1/x sinon
4) IF (NOC = 2) AND (S > 800) THEN return true ELSE return false
5) $(x \wedge true) \rightarrow ((x \vee y) \vee (z \leftrightarrow (x \wedge y)))$
6) i =1; while (i < 20) { i = i +1 }
7) x=n !
8) n!=1 si n=0 et n ! =n*(n-1)! sinon
9) $2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$

Write the corresponding algorithm

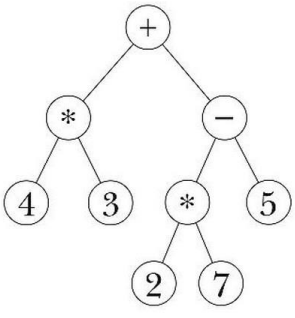
Exercise 2

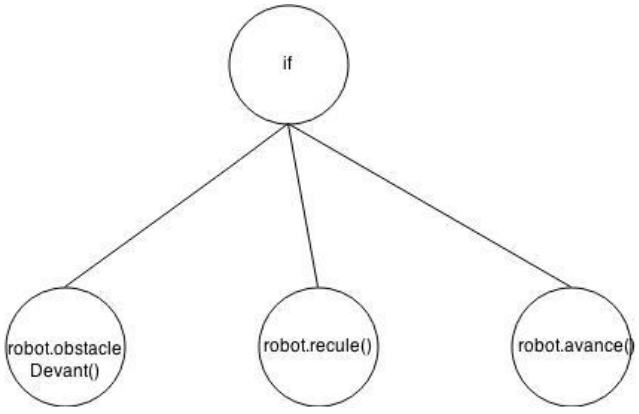
Write in java :

1) (+ 1 2 (IF (> TIME 10) 3 4))
2) (define (factorial x) (if (= x 0) 1 (* x (factorial (- x 1)))))
3) (* 2 (cos 0) (+ 4 6))
4) (defun fibonacci (N)
   "Compute the N'th Fibonacci number."
   (if (or (zerop N) (= N 1))
   1
   (+ (fibonacci (- N 1)) (fibonacci (- N 2)))))

Exercise 3

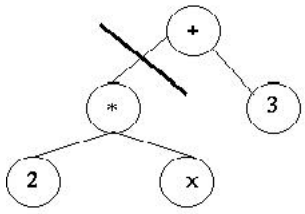
Find the corresponding s-expressions then write the corresponding algorithm.

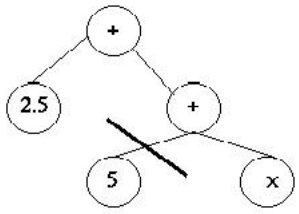




Exercise 4

Give the 2 child for the following parents crossover then write the corresponding algorithm:





Exercise 5

Let terminal set = {a,b,intrandom(1-5)}
Let function set = {+,*,sqrt}
Give an initial population of 6 individuals using :

- Grow approach ;
- Full approach ;
- Ramped half and half ;

Write the corresponding algorithm in each case.

Exercise 6 Even-parity-4 problem using GP. In its general formulation, the Boolean even-parity k function of k Boolean arguments returns true if an even number of its Boolean arguments evaluates true, otherwise it returns false. For the special case where k = 4, 16 fitness cases must be checked to evaluate the fitness of an individual.
Formulate the objective function for this problem. Our goal is to design a genetic programming approach to solve the problem. Propose a tree-like representation for this problem. For this purpose, the operators and the terminal set of the tree have to be defined.

Exercise 7 Distance measure for GP trees. Genetic programming use parse trees to encode solutions. Defining a distance between trees is not a trivial task.
Propose such a distance to deal with tree-based representations. As shown in the previous chapter, the distance measure may be important in the landscape analysis of the problem and the design of diversification or intensification search mechanisms. Is the distant proposed coherent with the mutation operators?

Exercise 8 Decision tree generation using GP. Decision trees represent a classical model in data mining for classification tasks. The problem consists in the induction using GP of the "optimal" decision tree from a training set (or learning set) of data. Each record from the training set is composed of several attributes and one goal attribute that is the class attribute. The decision tree will be used to predict the class of new records with an unknown class (label).
A decision tree is a tree where the terminal nodes are defined by classes ci, i = 1, n, and the nonterminal nodes represent the different attributes aj, j = 1,m. Each edge of the tree is defined by a test on the attribute values aj = vj where vj represents the possible values of the attribute aj.
Design a GP algorithm for this problem.

Exercise 9

Retrieve the function f (x) = 0,75x2-5x + 3 (values chosen completely randomly)
knowing 25 points of the function distributed in the interval [0,10]

objective	To determine a program that approximates the function f (x) = 0.75x2-5x + 3, we will not give any help to the algorithm to facilitate its search (ie the algorithm will not look for a polynomial and will not know the degree of the higher term)
Terminal set	x or float constant
functions	+, -, *
Regression set	25 values between 0 et 10
Fitness	Mean absolute difference or deviation
Stopping criterion	Fitness < = 1,25
Parameters	1000 individuals , 100 generations

Références bibliographiques :

- Brameier, M. (2004), *On Linear Genetic Programming*
- Koza, J.R. (1990), Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems, Stanford University Computer Science Department technical report STAN-CS-90-1314. A thorough report, possibly used as a draft to his 1992 book.
- Koza, J.R. (1992), Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press
- Koza, J.R. (1994), Genetic Programming II: Automatic Discovery of Reusable Programs, MIT Press
- Koza, J.R., Bennett, F.H., Andre, D., and Keane, M.A. (1999), Genetic Programming III: Darwinian Invention and Problem Solving, Morgan Kaufmann
- Koza, J.R., Keane, M.A., Streeter, M.J., Mydlowec, W., Yu, J., Lanza, G. (2003), Genetic Programming IV: Routine Human-Competitive Machine Intelligence, Kluwer Academic Publishers
- Langdon, W. B., Poli, R. (2002), *Foundations of Genetic Programming*, Springer-Verlag
- Poli, R., Langdon, W. B., McPhee, N. F. (2008), *A Field Guide to Genetic Programming*, freely available via Lulu.com.
- Weise, T., Global Optimization Algorithms - Theory and Application

Master 2 IDO 2019/2020

Optimization methods

Recitation 3 / ACO

Exercise 1

1) Rewrite the ACO algorithm for solving TSP seen in the course.
2) Evaluate the complexity of this algorithm.
3) Run the two first iterations for an instance of n=4 cities A, B, C, D such as :
   AB=6 AC=5 AD=8 BC=4 BD=4 CD=5
   With each of the following parameter configuration:
   a) Q=20, α=1, β=1 , τ0=1 ;
   b) Q=20, α=1, β=5 , τ0=1 ;
   c) Q=20, α=5, β=1 , τ0=1 ;
   Take ρ = 0.01 , ρ = 0.01 puis ρ = 0.1. What can we conclude?

4) Modify the algorithm for the versions:
   a) Elitist with e = n/10 ;
   b) MMAS for taumin= 0.1 and taumax=1 ;

Exercise 2

Define a distance function in order to design ACO algorithm for solving each of the following problems:

a) Simple KSP
b) SAT-3 ;
c) 8 Queens ;
d) Graph coloring ;
e) $1 \mid\mid \sum_{j=1}^{n} T_j$;
f) Multidimensionnel KSP ;
g) Multiple KSP ;

Exercise 3.

ACO design for a permutation scheduling problem.

The single machine total weighted tardiness problem (SMTWTP) is a well-known NP-hard scheduling problem. Given n jobs to be scheduled on a single machine. Each job i is characterized by its weight w_i, a processing time p_i, and a due date d_i. For a given schedule, let C_i define the completion time of the job i. The objective of the SMTWTP problem is to minimize the total weighted tardiness:

$\sum_{i=1}^{n} w_i \times T_i$ where $T_i = \max(0, C_i - d_j)$ represents the tardiness for the job i.

The main problem-dependent elements in designing an ACO algorithm are the pheromone information and the solution construction. Propose a pheromone information that is suitable to the problem. How this pheromone information will be used in the solution construction? Notice that the relative position of the job is much more important than its direct predecessor or successor in

the schedule such as in the traveling salesman problem. In addition to the pheromone, a heuristic to guide the ants toward good solutions must be proposed. Which priority greedy heuristic may be used to decide the next job to schedule?

Exercise 4.

ACO for the set covering problem.

The set covering problem (SCP) represents an important class of NP-hard combinatorial optimization problems with many applications in different domains: transportation network, integer coefficients linear programming, assignment problems, simplification of Boolean expressions, and so on. The SCP problem is generally defined by its Boolean covering matrix: $(a_{ij})_m \cdot n$ where $i \in I = 1,...,m$ and $j \in J = 1,...,n$. An element of the matrix $a_{ij} = 1$ if the row i is covered by the column j, and $a_{ij} = 0$ otherwise. Each column j of the matrix is characterized by a positive cost c_j and a cardinality ($card_j$), which is the number of rows covered by the column j. The SCP problem consists in choosing a subset of columns at a minimal cost in a way to cover all the rows (at least a 1 on each line). The SCP can be stated as minimizing $\sum_{j=1}^{n} c_j x_j$ such as: $j = \begin{cases} 1 & if\ the\ column\ j\ belong\ to\ the\ solution\ x \\ 0 & otherwise \end{cases}$

$$\sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m$$

$$x_j \in \{0, 1\}, j = 1, \ldots, n$$

The constraint maintains the integrity of the system, that is, each row is covered at least by one column. The parameter α_i is defined as the set of columns that cover the line i and β_j and as the set of lines covered by the column j: $\alpha_i = \{j \in J / a_{ij} = 1\}$ $\beta j = \{i \in I | aij = 1\}$

Propose a greedy algorithm to solve the SCP problem. Based on this greedy algorithm, design a solution construction strategy for the ACO algorithm. Once a covering has been established by an ant, an iterative function that eliminates the redundant columns is performed, that is, those that are useless (superfluous) since their elements are covered by other columns of the covering. This function iterates on the columns, beginning from the last one whose cost is more significant, to optimize the cost.

Exercise 5.

ACO for the shortest common super sequence (SCS) problem.

String problems are one of the most important and well-studied problems in computer science. Such an example of string problems is the SCS problem. Given an alphabet and a set L of strings over this alphabet, the SCS problem consists in finding a minimal length of super sequence of each string of L. A string S is a super sequence of a string A if S can be obtained from A by inserting in A zero or more characters. For instance, if the set L is {bbbaaa,bbaaab,cbaab,cbaaa}, the optimal super sequence, the one minimizing its length, will be cbbbaaab. The SCS problem has many applications in various domains:

- **Phylogenetics in bioinformatics:** The strings represent DNA from different species. One of the main questions concerning the evolution of those species is what will be their ancestor. For simplicity, let us assume that only mutations based on deletion of nucleotides can occur during the evolution process. Then, most probably the ancestor is the shortest super sequence of the input DNA strings.
- **Conveyer belt design:** Given a set of work pieces $\{W_1,W_2,...,W_n\}$ to be designed. To each work piece W_i is associated a sequence of operations for designing it, $W_i = s_1s_2 ...s_k$. The problem

is to produce a conveyer belt on which the n work pieces can be designed. Such a conveyer belt can be defined as a super sequence. The cost of producing the conveyer belt is mainly related to the size of the string representing its sequence. For instance, let us have three work pieces $\{W_1,W_2,W_3\}$ and their associated sequences to produce them: W_1 = bcbacb, caac, bbacac. A conveyer belt on which the three work pieces are designed can be bbacbaacb. Figure 3.66 presents an example of a super sequence for the conveyer belt.Ant can construct solutions by iteratively removing characters from the front of the input strings of L, and appending them to the super sequence. Then, each ant

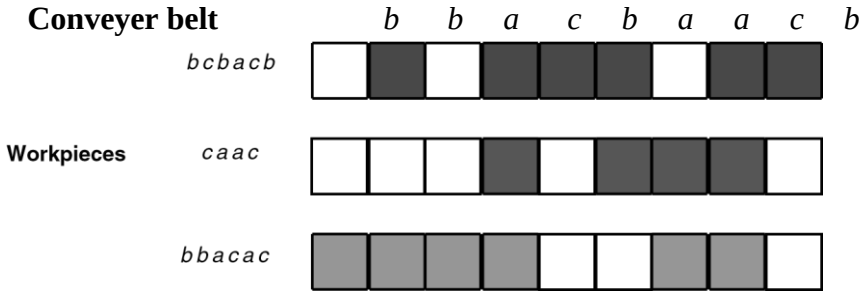


FIGURE 7.

Design of a conveyer belt to produce work pieces. A feasible non optimal solution is shown. The optimal solution has a size of 8 : bcbacacb must maintain a set of pointers to the current front of the strings that represent the selectable components of the constructed solution. The transitions are defined by the rules that select the next character of the super sequence. Only feasible super sequences are generated. Design an ACO algorithm for this problem.

REFERENCES

- E. Bonabeau, M. Dorigo, G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems, 1999
- M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem, 1997.
- M. Dorigo and T. Stützle. Ant Colony Optimization, MIT. Press, 2004.
- J. Kennedy and R. Eberhart. Swarm Intelligence, 2001.
- http://www.cnrs.fr/Cnrspresse/n386/html/n386a09.htm

Formation master 2 IDO 2018/2019
Module méthodes d'optimisation
TD4 / optimisation par colonie d'abeilles

Exercice 1

1) Réécrire le pseudo code de l'algorithme général d'optimisation par colonie d'abeilles. Appliqué au PVC.
2) Evaluer la complexité de cet algorithme.
3) Expliquer l'impact de chacun des paramètres suivant sur l'efficacité de l'algorithme :
   - Nombre d'abeilles scouts ;
   - Nombre d'abeilles élitistes ;
   - Nombre d'abeilles pour les sites riches en nectar ;
   - Nombre d'abeilles pour les sites pauvres en nectar ;
   - Taille du voisinage.

Exercice 2

Ecrire un programme simple pour implémenter une version algorithme d'OCA dans le but de trouver le minimum de la fonction de test de De Jong f (x) = $\sum_{i=1}^{n} x_i^2$
(Implémenter toutes les phases de l'algorithme)

Exercice 3

On désire fabriquer le même lot de N pièces de voiture par K différentes unités de production, où la production de chaque pièce par l'unité i nécessite $t_i > 0$ unités de temps, i = 1,2, ..., K.
Essayez d'allouer la charge de travail pour que la production des pièces soit achevée le plus tôt possible.
Formuler le problème puis proposer un algorithme d'OCA pour le résoudre.

Exercice 4

Dans un algorithme d'OCA, les butineuses actives recherchent dans le voisinage de la source précédente xi de nouvelles sources vi ayant plus de nectar, Elles calculent ensuite leur fitness.
Afin de produire une nouvelle source de nourriture à partir de l'ancienne, on utilise la formule :

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})$$

Où $k \in \{1, 2, \dots, Bn\}(Bn\ est\ le\ nombes\ des\ butineuses\ actives) et\ j \in \{1, 2, \dots, Sn\}$ sont des indices choisis au hasard. Bien que k est déterminé aléatoirement, il doit être différent de i. ϕ_{ij} est un nombre aléatoire appartenant à l'intervalle [-1, 1], il contrôle la production d'une source de nourriture dans le voisinage de xij .
Après la découverte de chaque nouvelle source de nourriture v_{ij}, un mécanisme de sélection gourmande est adopté, c'est-à-dire que cette source est évaluée par les abeilles artificielles, sa performance est comparée à celle de x_{ij}. Si le nectar de cette source est égal ou meilleur que celui de la source précédente, celle-ci est remplacée par la nouvelle. Dans le cas contraire l'ancienne est conservée.

Pour un problème de minimisation, La fitness est calculée suivant la formule :

$$fit_i(\overrightarrow{x_i}) = \begin{cases} \dfrac{1}{1 + f_i(\overrightarrow{x_i})} & si\ f_i(\overrightarrow{x_i}) \geq 0 \\ 1 + abs\left(f_i(\overrightarrow{x_i})\right) & si\ f_i(\overrightarrow{x_i}) > 0 \end{cases}$$

Telle que $f_i(\overrightarrow{x_i})$est la valeur de la fonction objectif de la solution xi.

A ce stade, les butineuses inactives et les éclaireuses qui sont en train d'attendre au sein de la ruche. A la fin du processus de recherche, les butineuses actives partagent les informations sur le nectar des sources de nourriture ainsi que leurs localisations avec les autre abeilles via la danse frétillante. Ces dernières évaluent ces informations tirées de toutes les butineuses actives, et choisissent les sources de nourriture en fonction de la valeur de probabilité P_i associée à cette source, et calculée par la formule suivante :

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n}$$

Où fit_i est la fitness de la solution i, qui est proportionnelle à la quantité du nectar de la source de nourriture de la position i).

La source de nourriture dont le nectar est abandonné par les abeilles, les éclaireuses la remplacent par une nouvelle source. Si durant un nombre de cycle prédéterminé appelé « limite » une position ne peut être améliorée, alors cette source de nourriture est supposée être abandonnée.

Utiliser ces formules pour dérouler manuellement l'algorithme d'OCA appliqué à l'optimisation de la fonction $f(x) = -x^2 + 4x$ dans l'intervalle [1 , 3] avec une précision de 1/10 en utilisant les paramètres suivants:

Nombre d'abeilles n =8 ;
Nombre de meilleurs sites m=5 ;
Nombre d'abeilles élitistes e = 3 ;
Recrutement riche n2 =2 ;
Recrutement pauvre n1=1 ;
Taille du voisinage ngh = 1 .

Formation master 2 IDO 2018/2019
Module méthodes émergentes
TD5 / PSO

Exercice 1

Consider an illustrative example of a particle swarm optimization system composed of three particles and $V_{max} = 10$. To facilitate calculation, we will ignore the fact that r_1 and r_2 are random numbers and fix them to 0.5 for this exercise. The space of solutions is the two dimensional real valued space R^2 and the current state of the swarm is as follows:
Position of particles: $x_1 = (5,5)$; $x_2 = (8,3)$; $x_3 = (6,7)$;
Individual best positions: $x_1^* = (5,5)$; $x_2^* = (7,3)$; $x_3^* = (5,6)$;
Social best position: $x^* = (5,5)$;
Velocities: $v_1 = (2,2)$; $v_2 = (3,3)$; $v_3 = (4,4)$.
Please answer the following questions:
1. What would be the next position of each particle after one iteration of the PSO algorithm using inertia $\omega = 1$?
2. And using $\omega = 0{:}1$?
3. Explain why the parameter ω is called inertia.
4. Give an advantage and a disadvantage of a high inertia value.

Exercice 2

1. Appliquer la méthode PSO à l'exemple simple $F(x) = x^2$ et avec $c_1 = c_2 = c_3 = 0.7$.
2. Tracer les trajectoires correspondantes des particules sur la parabole.
3. Appliquer la méthode PSO au cas de la fonction de Rastrigin sur $[-5; 5]^2$:

$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2)$ avec 10 particules. Qu'observe t-on?

Exercise 3

Réécrire l'algorithme PSO et discuter sa complexité.

Exercise 4

Appliquer PSO au TSP: interpréster:

- Une particule ;
- Une vélocité ;
- Une différence de positions ;
- Une somme de vélocités ;
- Les deux équations de mise à jour.
- Donner des exemples.

Meme question pour KSP et SAT-3.

Exercice 5

A Particle Swarm Optimization (PSO) is applied to the search of a hidden RF transmitter. The search area is a square stretched out between coordinates $(-100, -100)$ and $(100,100)$. The amplitude of a sampled signal as a function of distance from the emitter is given by:

$$A = \frac{1}{4\pi r^2} + kN(0, \sigma)$$

where A is the measured amplitude, r is distance between transmitter and the sampling location and N is a Gaussian noise distribution with zero mean and σ standard deviation, k is a parameter for adjusting the relative noise level.

a. Explain the canonical PSO.

b. Given 4 particles in a PSO with positions:

$x_1 = (10,10)$
$x_2 = (12,8)$
$x_3 = (11, -10)$
$x_4 = (-4,9)$

Calculate an iteration of particle 1 assuming $\omega = 0.98, \omega_1 = 0.04, \omega_2 = 0.02$ and simulate the required probabilities. Also, assume that the position of the hidden emitter is $(0,0)$ and that $k = 0$, i.e. a noise free system.

c. Simulate the next iterations of this PSO problem by altering the the NetLogo version of PSO, found under:
'File->Models Library->Sample Models->Computer Science->Particle Swarm Optimization'.
Remark, the UpdateParticleVelocity (the 'to go' function) in the NetLogo program is altered. Also, you could use new random initial position and velocities for the particles.

d. Release the 4 particles from (-75,-75) plus some randomness. How does this affect the optimization?

e. What happen if you add noise to the system? You could set $k = 0{,}0001$. Compare the two different initial positions of the particles. Would you use PSO in a real swarm robotic system where the mobile robots are released from same location?

f. What could be alternative approaches for locating the hidden RF emitter using swarm robotics?

g. Optional:
Play with different PSOs, parameters and possibly other swarm algorithms on this problem.

Master 1 IDO 2020/2021

Optimisation combinatoire 1

TD1 / Complexité des algorithmes

Exercice 1

Exprimer les fonctions suivantes en notation O , Ω puis θ :

$f_1(n)=3n$, $f_2(n)=2^n$ $f_3(n)=n^2$, $f_5(n)=n^n$,

$f_6(n)=\log n$, $f_7(n)=n!$, $f_8(n)= n\log n$

Exercice 2

Pour chacun des fonctions $T_i(n)$ suivant, déterminer sa complexité asymptotique dans la notation Grand-O.

$T_0(n) = 3n$

$T_1(n) = 6n^3 + 10n^2 + 5n + 2$

$T_2(n) = 3\log_2 n + 4$

$T_3(n) = 2^n + 6n^2 + 7n$

$T_4(n) = 7k + 2$

$T_4(n) = 4\log_2 n + n$

$T_5(n) = 2\log_{10} k + kn^2$

Exercice 3

Considérer les deux algorithmes A1 et A2 avec leurs temps d'exécution $T_1(n) = 9n^2$ et $T_2(n) = 100n + 96$ respectives.

1. Déterminer la complexité asymptotique des deux algorithmes dans la notation Grand-O. Quel algorithme a la meilleure complexité asymptotique ?
2. Montrer que vos solutions sont correctes en spécifiant un c et un n_0 par algorithme afin que la relation suivante soit satisfaite : $O(f) = \{g | \exists c > 0 : \exists n_0 > 0 : \forall n \geq n_0 : g(n) \leq cf(n)\}$
3. Calculer les temps maximales d'exécution des deux algorithmes $T_i(n)$ pour $n = 1, n = 3, n = 5, n = 10, n = 14$.
4. Ebaucher les graphes des deux fonctions T_i dans un même système de coordonnés (abscisse n, ordonné $T_i(n)$).
5. Etudier quel algorithme est le plus efficace en fonction de n?
6. Quelle est la complexité asymptotique de l'algorithme suivant ? Quelle règle avez vous appliqué ?

début
    appeler A1
    appeler A2
fin

Exercice 4

Addition de matrices

Considérer les deux matrices quadratiques A et B de taille n :

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}, B = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{pmatrix},$$

L'addition de ces deux matrices donne la matrice C quadratique de taille n:

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}$$

Avec *cij = aij + bij* $\forall i,\forall j$

1. Définir le type des matrices quadratiques et déclarer les variables A,B, et C.
2. Ecrire un algorithme qui effectue l'addition des deux matrices A et B et stocke les résultats en C.
3. Déterminer la fonction de temps maximale (''worst case'') T(n) pour des matrices de taille *n*.
4. Déterminer la complexité Grand-O pour des matrices de taille n.

Exercice 5

Multiplication de matrices

La multiplication des deux matrices quadratiques de taille n donne la matrice C quadratique de taille n:

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix}$$

avec

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \quad \forall i, \forall j$$

1. Ecrire un algorithme qui effectue la multiplication des deux matrices A et B et stocke les résultats en C.
2. Déterminer la fonction de temps maximale (''worst case'') T(n) pour des matrices de taille n.
3. Déterminer la complexité O(n) pour des matrices de taille *n*.

Exercice 6

Écrire l'algorithme qui recherche un élément dans un vecteur de taille n. Calculer la complexité temporelle en fonction du nombre de comparaisons dans le pire et dans le meilleur des cas. Refaire les calculs en fonction du nombre d'accès au vecteur.

Exercice 7

Écrire l'algorithme de tri par sélection. Calculer la complexité temporelle en fonction de nombre de comparaisons et de permutations dans le pire et dans le meilleur des cas. Refaire les calculs en fonction du nombre d'accès au vecteur.

Exercice 8

On considère deux manières de représenter ce que l'on appelle des « matrices creuses », c'est-à-dire des matrices d'entiers contenant environ 90% d'éléments nuls :

a) La matrice est représentée par un tableau à deux dimensions dont les cases contiennent les éléments.
b) La matrice est représentée par un tableau à une dimension. On ne s'intéresse qu'aux éléments de la matrice *qui ne sont pas nuls*. Chaque case du tableau contient un triplet (i, j, a) correspondant à l'indice de ligne, l'indice de colonne, et la valeur d'un élément non nul.

Le problème considéré consiste à calculer la somme des éléments d'une matrice. On demande d'écrire un algorithme permettant de calculer cette somme, pour chacune des deux représentations, puis de comparer leur complexité spatiale (espace mémoire occupé) et leur complexité temporelle (nombre d'opérations à effectuer). Que peut-on conclure de cette comparaison ? Montrer qu'il

existe une valeur critique du nombre d'éléments non nuls à partir de laquelle une méthode l'emporte sur l'autre.

Exercice 9

Pour chacun des algorithmes suivants évaluer le nombre d'opérations :

```
Algo 1 affichage des n composantes du vecteur x
Pour i allant de 1 à n faire
      afficher(xi)
Algo 2
Pour i allant de 1 à n faire
   Pour j allant de 1 à n faire
       afficher(xi+xj)
Algo 3
Pour i allant de 1 à n faire
 Pour j allant de 1 à n faire
    Pour k allant de 1 à n faire
          Pour l allant de 1 à n faire
                Pour m allant de 1 à n faire
                      afficher(xi+xj+xk+xl+xm)
```

Exercice 10

On considère, pour effectuer la recherche d'un élément dans un tableau, la recherche séquentielle et la recherche dichotomique. On s'intéresse à leur complexité temporelle.

Pour cela, considérer un tableau ayant mille éléments (version trié, et version non trié). Pour chaque algorithme, et pour chaque version du tableau, combien de comparaisons sont à effectuer pour :

- trouver un élément qui y figure ?
- trouver un élément qui n'y figure pas ?

Quels sont les cas où le tableau est parcouru complètement et les cas où un parcours partiel est suffisant ? Conclure en donnant la complexité temporelle pour chaque algorithme

Exercice 11

Soient les 3 algorithmes suivants permettant de calculer la valeur d'un polynôme
$p(x) = a0+a1x+a2x^2+...+anx^n$
en un point x. Comparer leurs complexités.

```
algo 1
début
      p = a[0] ;
      pour i := 1  à  n  faire
            calculer p+ = a[i]* x^i ;
algo 2
p = a[0] ;
q = 1;
pour i = 1  à  n  faire
     q = q * x ;
     p := p  +  a[i] * q ;

algo 3
p = a[n] ;
pour i := n  à  1, pas -1, faire
      p := p*x + a[i-1] ;
```

Exercice 12

On considère un tableau à une dimension contenant des lettres majuscules. On désire compter la fréquence de chacune des 26 lettres de l'alphabet. Ecrire deux procédures qui donnent en sortie un tableau de fréquence: l'une où le tableau est parcouru 26 fois, et l'autre (plus performante !) où le calcul est fait en un seul parcours. On pourra supposer que l'on dispose d'une fonction auxiliaire *position(lettre)* qui pour chaque lettre donne sa position dans l'alphabet : *position('A') = 1, ..., position('Z) = 26.*

Exercice 13

Soit l'algorithme suivant qui effectue la recherche dichotomique du rang (place) d'un nombre A dans une suite triée (par ordre croissant) de n nombres mis dans un tableau à une dimension (vecteur L[i]; i=1,..,n). Cet algorithme fonctionne sous l'hypothèse que A est présent dans la liste.

Algorithme :
début
place = 1 ;
f = n ;
tant que place < f **faire** milieu = (place + f) / 2 ; **si** L[milieu] < A **alors** place = milieu + 1 ;
sinon f = milieu ;
finsi; **fait**; **fin**;

Question Donner la complexité de cet algorithme.

Exercice 14

Écrire l'algorithme de tri à Bulles. Calculer la complexité temporelle en fonction de nombre de comparaisons et de permutations dans le pire et dans le meilleur des cas. Refaire les calculs en fonction du nombre d'accès au vecteur. Comparer avec le tri par sélection.

Exercice 15

Calculer la complexité temporelle de l'algorithme de recherche dichotomique en fonction du nombre de comparaisons dans le pire et dans le meilleur des cas. Refaire les calculs en fonction du nombre d'accès au vecteur. Comparer avec l'algorithme de recherche séquentiel.

Exercice 16

On considère trois tris élémentaires : le tri sélection, le tri par insertion (trois variantes), et le tri bulle. On considère pour un tableau les deux cas extrêmes où le tableau est déjà trié (dans l'ordre croissant), et celui où il est trié dans l'ordre décroissant. Décrire avec précision le comportement et la complexité de chacun des algorithmes dans ces deux cas. Quelles conséquences peut-on en tirer ?

Exercice 17

Compléter le tableau ci-dessous

Nombre d'opérations en fonction de la taille n des données	**Avec un ordinateur X**		**Avec un ordinateur Y 100 fois plus rapide que X**	
	Taille maximale (n max) des problèmes traités en 1h	Nombre d'opérations effectuées en 1h	Taille maximale (n max) des problèmes traités en 1h	Nombre d'opérations effectuées en 1h
5n				
Log n				
n^2				
2^n				

Master 1 IDO 2020/2021

Optimisation combinatoire 1

TD 2 / Eléments d'optimisation combinatoire

Exercice 1

Problème du voyageur de commerce

Un ordinateur a une variété de composants à connecter par des fils. La distance en millimètres entre chaque paire de composants est donnée dans le tableau ci-dessous. Déterminez les paires de composants à connecter afin que l'ensemble des composants soit connecté et que la longueur totale du fil entre les composants soit minimisée.

	1	2	3	4	5	6
1	0	6,7	5.2	2,8	5,6	3,6
2	6,7	0	5,7	7,3	5.1	3.2
3	5.2	5,7	0	3.4	8.4	4.0
4	2,8	7,3	3.4	0	8,0	4.4
5	5,6	5.1	8.4	8,0	0	4.6
6	3,6	3.2	4.0	4.4	4.6	0

Exercice 2

Planification de la production à produits semi-finis

L'entreprise fabrique les produits P_1, P_2 et P_3.

Pour produire 1 unité de produit P_1, l'entreprise utilise 3 kg de matière.

Pour produire 1 unité de produit P_2, l'entreprise utilise 2 kg de matière et 1 unité de produit P_1.

Pour produire 1 unité de produit P_3, l'entreprise utilise 2 kg de matière, 2 unités de produit P_1 et 1 unité de produit P_2. Il y a 1000 kg de matériel disponible.

Les produits P_1 et P_2 utilisés comme produits semi-finis peuvent également être vendus eux-mêmes. Les prix des biens P_1, P_2 et P_3 sont de 5, 10 et 30 e. L'objectif est de maximiser les revenus totaux des produits vendus. Formulez un modèle mathématique du problème.

Exercice 3

Problème de coupe de stock

L'entreprise produit des clôtures en lattes de jardin. Il n'y a que des lattes standard de 200 cm de long à disposition dans l'entrepôt. Pour produire une clôture, l'entreprise a besoin d'exactement 1200 lattes de 80 cm de long, 3100 lattes de 50 cm de long et 2100 lattes de 30 cm de long.

Vous devez concevoir un plan de coupe pour minimiser la quantité totale de lattes de 200 cm de long.

Formulez un modèle mathématique du problème.

Exercice 4

Problème du sac à dos

Il y a 5 projets caractérisés par le coût d'investissement et le rendement. Le budget de 50 000 est disponible pour sélectionner les projets qui assurent le rendement total le plus élevé.

	P1	P2	P3	P4	P5
Coût	12 000	10 000	15 000	18 000	16 000
Revenu	20 000	18 000	22 000	26 000	21 000

Exercice 5

Problème de correspondance parfaite

10 étudiants partent en voyage scolaire. Pour les attribuer à des chambres doubles, on leur a demandé d'exprimer leurs préférences (voir le tableau, 0 min, 10 max). Pour $i < j$, la valeur c_{ij} est la valeur de préférence exprimant l'étudiant i veut être dans la pièce avec l'étudiant j , pour $i > j$, la valeur c_{ij} est la valeur de préférence exprimant l'étudiant j veut être dans la pièce avec l'étudiant i . Affecterez-vous des étudiants dans des salles pour maximiser le bonheur total du groupe ?

Exercice 6

Problème d'affectation linéaire

Une course de relais pour des équipes de 5 membres est organisée. Un membre de chaque équipe concourra dans une discipline. Vous allez constituer une équipe la plus solide. Dans le tableau, les meilleures performances saisonnières (en minutes) des candidats sont données.

SB	Courir	Nager	Bicyclette	En ligne	Ski
1	75	25	202	130	165
2	87	24	198	127	173
3	68	19	195	121	164
4	91	20	207	122	182
5	80	28	215	125	172
6	78	22	197	125	180
7	75	25	205	127	178
8	81	23	211	131	165

Exercice 7

Problème d'affectation de goulot de bouteille

Le projet se compose de 5 parties indépendantes. Dans l'entreprise, 5 départements peuvent gérer les pièces individuellement. Les données historiques montrent le temps moyen (en jours) que les départements ont terminé des tâches similaires (voir le tableau). NA représente le fait qu'un ministère n'a pas travaillé sur une telle tâche dans le passé. L'entreprise souhaite terminer l'ensemble du projet le plus rapidement possible.

Temps	Partie 1	Partie 2	Partie3	Partie 4	Partie5
Dépt1	25	15	N / A	17	25
Dépt2	22	N / A	22	20	22
Dépt3	20	18	25	16	23
Dépt4	N / A	20	30	21	28
Dépt5	27	19	27	18	N / A

Exercice 8

Problème d'emplacement d'installations

L'entreprise utilise 7 entrepôts potentiels pour ses 5 filiales. Dans le tableau ci-dessous, les besoins mensuels des filiales et les capacités mensuelles des entrepôts sont indiqués (en milliers de tonnes). Si un entrepôt est utilisé, l'entreprise doit payer un loyer mensuel (en milliers). De plus, le coût unitaire de transport (par tonne) est calculé pour chaque paire d'entrepôt et filiale. Quel entrepôt utiliser et quelles quantités de matériel transporter entre les entrepôts et les filiales. L'objectif est de minimiser le coût mensuel total.

FLP	SD1	SD2	SD3	SD4	SD5	Cap	Rent

WH1	10	15	20	12	8	20	10
WH2	7	10	15	22	13	25	12
WH3	20	13	10	11	9	15	8
WH4	15	12	21	18	16	18	9
WH5	11	22	12	10	15	22	11
WH6	9	13	11	18	22	30	13
WH7	18	10	15	7	9	23	11
Req	25	22	17	22	15		

Exercice 9

Problème d'affectation quadratique

L'entreprise a l'intention de créer 5 entrepôts dans 5 villes. Dans le premier tableau, les distances (en km) entre les villes sont données. Le deuxième tableau montre un certain nombre de déplacements nécessaires entre les entrepôts dans un délai d'un mois. L'objectif est d'allouer les entrepôts en minimisant le coût total de déplacement.

Voyages	WH1	WH2	WH3	WH4	WH5
WH1	0	10	15	12	8
WH2	9	0	18	16	10
WH3	20	8	0	10	12
WH4	10	15	11	0	22
WH5	17	12	9	11	0

Distance	Ville1	Ville2	Ville3	Ville4	Ville5
Ville1	0	50	60	130	100
Ville2	50	0	70	150	120
Ville3	60	70	0	80	40
Ville4	130	150	80	0	50
Ville5	100	120	40	50	0

EXERCICE 10

Problème de rangement

Les produits doivent être transportés chez le client dans des conteneurs identiques. Dans le tableau, un poids unitaire de chaque type de produit (en kg) et un nombre d'entre eux à transporter sont indiqués. La capacité de poids du conteneur est de 500 kg. L'objectif est de minimiser le nombre de conteneurs usagés.

Bin Packing	Poids	Nombre
Produit1	20	13
Produit2	22	15
Produit3	18	25
Produit4	15	30
Produit5	21	18
Produit6	16	35

EXERCICE 11

Problème de débit maximum

Trouver le débit maximum du nœud 1 au nœud 6 pour le graphe donné par le tableau suivant

Arc	Capacité	Arc	Capacité
(1,2)	10	(3,5)	7
(1,3)	10	(3,6)	5
(1,4)	12	(4,3)	3
(2,5)	11	(4,6)	9
(3,4)	3	(5,6)	18

EXERCICE 12

Trouver le flux (de 1 à 6) de valeur 25 avec le coût total minimal. Dans le tableau, la capacité et le coût unitaire de chaque arc sont indiqués.

Arc	Capacité	Coût	Arc	Capacité	Coût
(1,2)	10	5	(3,5)	7	6
(1,3)	10	10	(3,6)	5	9
(1,4)	12	20	(4,3)	3	12
(2,5)	11	11	(4,6)	9	17
(3,4)	3	12	(5,6)	18	8

EXERCICE 13

Problème de débit maximal limité au coût

Soit 700 le budget du flux. Trouverez-vous le débit maximum de 1 à 6 en respectant cette restriction ?

Arc	Capacité	Coût	Arc	Capacité	Coût
(1,2)	10	5	(3,5)	7	6
(1,3)	10	10	(3,6)	5	9
(1,4)	12	20	(4,3)	3	12
(2,5)	11	11	(4,6)	9	17
(3,4)	3	12	(5,6)	18	8

EXERCICE 14

Problème de transbordement

Il est nécessaire de transporter des conteneurs vides des sources aux destinations. Dans le graphique, les nœuds 1 et 3 sont des sources avec des conteneurs d'approvisionnement 15 et 10, les nœuds 4 et 6 sont des destinations avec des conteneurs de demande 5 et 20. L'objectif est de minimiser le coût total.

Arc	Capacité	Coût	Arc	Capacité	Coût
(1,2)	10	5	(3,5)	7	6
(1,3)	10	10	(3,6)	5	9
(1,4)	12	20	(4,3)	3	12
(2,5)	11	11	(4,6)	9	17
(3,4)	3	12	(5,6)	18	8

EXERCICE 15

Arbre couvrant minimal

L'entreprise doit installer 6 panneaux d'information dans le parc de la ville. Ils doivent être reliés par un câble passant sous les trottoirs. Les distances (en 10 mètres) entre les planches sont indiquées dans le tableau. S'il n'y a pas de chaussée entre une paire de planches, une valeur prohibitive 100 est définie. L'objectif est de minimiser le coût total des travaux d'excavation et du câble lui-même.

Planches	1	2	3	4	5	6
1	0	6	5	100	100	100
2	6	0	7	2	4	100
3	5	7	0	6	100	8
4	100	2	6	0	3	4
5	100	4	100	3	0	5
6	100	100	8	4	5	0

EXERCICE 16

Arbre de Steiner minimal

Trois utilisateurs (nœuds 2, 3 et 4) doivent être connectés à l'émetteur (nœud 1) soit directement, soit via deux stations de transfert (nœuds 5 et 6). Dans le tableau, les valeurs de coût (en milliers d'e par mois) pour les connexions possibles sont données. L'utilisation des stations de transfert est facturée 30 et 20 milliers par mois. Trouver la valeur optimale de connexion.

Arc	Coût	Arc	Coût
(2,1)	15	(4,5)	9
(2,5)	3	(4,6)	6
(3,1)	18	(5,1)	7
(3,5)	4	(6,1)	12
(3,6)	7		

EXERCICE 17

Problème du voyageur de commerce

Un représentant commercial d'une brasserie doit visiter 7 pubs dans 7 villes. Dans le tableau suivant, les distances (en km) correspondent aux liaisons directes (routes) entre les villes. Un tiret indique qu'il n'y a pas de route directe entre les villes. L'objectif est de visiter tous les pubs en minimisant la durée totale de la visite.

	1	2	3	4	5	6	7	8
1	0	8	-	13	10	-	12	9
2	8	0	6	16	-	-	-	4
3	-	6	0	-	-	-	-	-
4	13	16	-	0	7	-	-	-
5	10	-	-	7	0	7	13	-
6	-	-	-	-	7	0	15	-
7	12	-	-	-	13	15	0	13
8	9	4	-	-	-	-	13	0

Le tableau suivant contient les distances entre toutes les paires de villes.

Distance	1	2	3	4	5	6	7	8
1	0	8	14	13	10	17	12	9
2	8	0	6	16	18	25	17	4
3	14	6	0	22	24	31	23	10
4	13	16	22	0	7	14	20	20
5	10	18	24	7	0	7	13	19
6	17	25	31	14	7	0	15	26
7	12	17	23	20	13	15	0	13
8	9	4	10	20	19	26	13	0

EXERCICE 18

Problème de routage de véhicules

Le représentant commercial de la brasserie (voir l'exemple 16) a conclu des contrats avantageux. Les pubs prendront des barils de bière dans les quantités indiquées dans le tableau suivant. Pour la livraison, un véhicule d'une capacité de 50 barils sera utilisé. L'objectif est de satisfaire toutes les exigences en minimisant la durée totale des circuits en véhicule.

	Exigence
1	0
2	18
3	10
4	15
5	12
6	10
7	8
8	11

Exercice 19

1. Le problème SUBSET SUM est donné comme suit
   INPUT: $n \in$ IN, n nombres naturels $a_1, \ldots, a_n$, nombre $B \in$ IN.
   QUESTION: Existe-t-il un sous-ensemble $I \subseteq \{1, \ldots, n\}$ tel que $\sum_{i \in I} a_i = B$ *soit* vrai?
   Le problème de partition PARTITION est donné comme suit
   INPUT: $n \in$ IN, n nombres naturels $a_1, \ldots, a_n$.
   QUESTION: Existe-t-il un sous-ensemble $I \subseteq \{1, \ldots, n\}$ tel que $\sum_{i \in I} a_i = \sum_{i / \in I} a_i$ est vrai?

(a) Montrer que SUBSET SUM est NP-complet. Vous pouvez supposer pour votre preuve qu'il est connu que PARTITION est NP-complete.

(b) Démontrez que PARTITION est NP-complète. Vous pouvez supposer pour votre preuve que SUBSET SUM est NP-complete.

2. Prouvez que le problème suivant est NP-difficile:
   INPUT: $n \in$ IN, $n \times n$ matrice $D = (d_{ij})$.
   QUESTION: Une tournée TSP d'une durée au plus (1+) fois la durée d'une tournée TSP la plus courte. (est une constante fixe > 0.)
   Vous pouvez supposer que l'on sait que le problème du cycle hamiltonien (HC) est NP-complet.
3. Prouvez que le problème de clique maximum (CLIQUE) est NP-complet.
   INPUT: Un graphe non orienté $G = (V, E)$, un nombre $k \in$ IN.

QUESTION: Existe-t-il une clique (sous-graphe complet de G) avec cardinalité (= nombre de sommets) $\geq k$?

Vous pouvez supposer que l'on sait que (VERTEX COVER) est NP-complet:

INPUT: Un graphe non orienté $G = (V, E)$ et un nombre $k \in$ IN.

QUESTION: Est-ce que G contient une couverture de sommet $C \subseteq V$ avec $\leq k$ sommets? (Un sous-ensemble V^0 de V est appelé couverture de vertex de G si chaque arête de E est incidente avec au moins un sommet de $V^{0.}$)

EXERCICE 20

Problème du facteur chinois non guidé

À l'Halloween, des enfants veulent visiter toutes les maisons du quartier (voir la figure). Les longueurs des rues (en mètres) qu'elles doivent traverser sont données dans le tableau. Allez-vous planifier une visite pour les enfants afin de minimiser la distance totale.

Arc	Longueur	Arc	Longueur
(1,2)	210	(6,7)	80
(1,9)	160	(6,11)	150
(2,3)	140	(7,8)	80
(2,5)	80	(7,9)	110
(3,4)	40	(9,10)	160
(3,5)	210	(10,11)	130
(4,6)	310	(10,12)	190
(5,6)	70	(11,12)	150

EXERCICE 21

Problème de facteur chinois guidé

Un véhicule qui ramasse les ordures des poubelles doit emprunter des rues à sens unique dans un quartier de Prague (voir la figure). Les longueurs des rues (en mètres) sont données dans le tableau. L'objectif est de minimiser la distance totale parcourue par le véhicule.

Arc	Longueur	Arc	Longueur
(1,2)	82	(7,12)	93
(2,3)	53	(8,4)	162
(2,6)	78	(8,7)	111
(3,7)	93	(9,5)	93
(4,3)	56	(9,11)	200
(5,1)	78	(10,9)	96
(6,5)	80	(11,10)	73
(6,10)	76	(11,12)	76
(7,6)	78	(12,8)	111

Master 1 IDO 2020/2021
Optimisation combinatoire 1
TD 3 / PROGRAMMATION DYNAMIQUE

Exercice 1
Triangle de Pascal

On veut calculer les coefficients binomiaux $C_n^k = \left(\begin{array}{c} n \\ k \end{array} \right) = \frac{n!}{k!(n-k)!}$.

Rappelons les propriétés suivantes :

$\left(\begin{array}{c} n \\ k \end{array} \right) = \left(\begin{array}{c} n-1 \\ k-1 \end{array} \right) + \left(\begin{array}{c} n-1 \\ k \end{array} \right)$ pour $0 < k < n$, $\left(\begin{array}{c} n \\ n \end{array} \right) = 1$ et $\left(\begin{array}{c} n \\ 0 \end{array} \right) = 1$.

1. Donner un algorithme récursif du calcul de $\left(\begin{array}{c} n \\ k \end{array} \right)$. Evaluer sa complexité.
2. Ecrire l'algorithme qui retourne $\left(\begin{array}{c} n \\ k \end{array} \right)$ en utilisant la technique de la programmation dynamique. Evaluer sa complexité.

Exercice 2
Problème du stockage
Considérons n programmes $P_1,P_2,...,P_n$ qui peuvent être stockés sur un disque dur de capacité D gigabytes.

- Chaque programme P_i a besoin de s_i gigabytes pour être stocké et a une valeur v_i
- Tous les programmes ne peuvent pas être stockés sur le disque : $\sum_{i=1}^{n} s_i > D$.

Les programmes stockés dans le disque dur doivent maximiser la valeur totale, sans dépasser la capacité du disque dur. L'objectif est de concevoir un algorithme qui permet de calculer un tel ensemble.
Nous allons construire un tableau T dans lequel les lignes seront indexées par les programmes et les colonnes par les valeurs. L´élément $T[i,j]$ représentera la valeur maximale pour un disque dur de capacité j à l'aide des i premiers programmes.

1. Donner la formule de récurrence.
2. Donner l'algorithme utilisant la programmation dynamique.
3. Donner la complexité de cet algorithme.

Exercice 3
Problème de chemin le plus long dans un graphe
Soit $G = (V,E)$ un graphe orienté avec $V = \{v_1,...,v_n\}$. On dit que G est ordonné s'il vérifie les deux propriétés suivantes :

1. Chaque arc de ce graphe est de la forme $(i \rightarrow j)$ si $i < j$
2. Tous les sommets sauf le sommet v_n ont au moins un arc sortant.

Ici, par souci de simplification, nous supposerons qu'il existe un chemin allant de v_i vers v_n pour tout $i = 1,...,n$.
L'objectif est de trouver le chemin le plus long entre les sommets v_1 et v_n.

1. Montrer que l'algorithme glouton suivant ne résout pas correctement le problème.

$u \leftarrow v_1$;
$L \leftarrow 0$;
Tant qu'il existe un arc sortant du sommet u
    choisir l'arc $(u \rightarrow v_j)$ tel que j est le plus petit possible
    $u \leftarrow v_j$;
    $L \leftarrow L + 1$;
retourner L

2. Donner la formule de récurrence qui permet de calculer la longueur du chemin le plus long commençant par v1 finissant par v`.
3. Donner un algo qui retourne la longueur du chemin le plus long commençant par v1 finissant par vn.
4. Modifier l'algorithme précédent afin qu'il retourne le chemin.
5. Donner la formule de récurrence permettant de calculer le chemin de poids maximum commençant par v1 finissant par vn.
6. En d´déduire l'algorithme.
7. Donner la formule de récurrence qui permet de calculer le chemin de poids maximal de ` arcs commençant par v1 finissant par vi . En déduire l'algorithme.

Exercice 4
Planning

Considérons un chef de projet qui doit gérer une équipe en lui affectant un projet qui dure une semaine. Le chef de projet doit choisir si il prend un projet stressant ou non stressant pour la semaine.

a) Si le chef de projet choisit le projet qui n'est pas stressant durant la semaine i, alors l'entreprise reçoit un revenu j.
b) Si le chef de projet choisit le projet qui est stressant durant la semaine i, alors l'entreprise recroit un revenu h_i et l'équipe ne travaille pas durant la semaine $i - 1$

Exemple : Si chef de projet choisit de se reposer à la semaine 1, puis de prendre le projet stressant à la semaine 2, puis de prendre les projets non-stressant à la semaine 3 et 4. Le revenu total est de 70 et il correspond au maximum.

	semaine 1	semaine 2	semaine 3	semaine 4
l	10	1	10	
h	5	50	5	
	1			

1. Montrer que l'algorithme suivant ne résout pas correctement le problème.

pour chaque itération $i = 1,...,n$
  si $h_{i+1} > l_i + l_{i+1}$ alors
    Choisir Ne pas travailler à la semaine i
    Choisir le projet stressant à la semaine i+1
    Continuer à l'itération $i + 2$
  sinon
    Choisir le projet non-stressant à la semaine i
    Continuer l'itération $i + 1$

2. Donner un algorithme qui retourne le revenu maximal que peut obtenir le chef de projet.

Exercice 5

Multiplications chainées de matrices.

On veut calculer le produit de matrices $M = M_1M_2 \cdots M_n$. Multiplier une matrice $p \times q$, par une matrice $q \times r$ en utilisant la méthode standard nécessite pqr produit scalaire.

1. Considérons 4 matrices A : 20×5, B : 5×100, C : 100×8, D : 5×30. On veut calculer le produit $ABCD$. En fonction des parenthétisations, le nombre de produits varie.

D´déterminer le nombre de produits pour calculer $ABCD$, si on utilise les parenthétisations suivantes : $((AB)C)D$ ou $(A(BC))D$

2. Ecrire une formule de récurrence pour calculer $c(i,j)$.
3. Ecrire un algorithme utilisant la programmation dynamique

Exercice 6

1. Donner un algorithme de programmation dynamique pour résoudre le problème suivant :
   Entrée : une matrice A de taille $n \times m$ où les coefficients valent 0 ou 1.
   Sortie : la largeur maximum K d'un carré de 1 dans A, ainsi que les coordonnées (I, J) du coin en haut à gauche d'un tel carré (autrement dit pour tout $i, j, I \leq i \leq I + K - 1, J \leq j \leq J + K - 1$,
   $A[i, j] = 1$).
2. Quelle est sa complexité ?

Exercice 7

La bibliothèque planifie son déménagement. Elle comprend une collection de n livres $b_1, b_2, \ldots, b_n$. Le livre b_i est de largeur w_i et de hauteur h_i. Les livres doivent être rangés dans l'ordre donné (par valeur de i croissante) sur des étagères identiques de largeur L.

1. On suppose que tous les livres ont la même hauteur $h = h_i, 1 \leq i \leq n$. Montrer que l'algorithme glouton qui range les livres côte à côte tant que c'est possible minimise le nombre d'étagères utilisées.
2. Maintenant les livres ont des hauteurs différentes, mais la hauteur entre les étagères peut se régler. Le critère à minimiser est alors l'encombrement, défini comme la somme des hauteurs du plus grand livre de chaque étagère utilisée.
   (a) Donner un exemple où l'algorithme glouton précédent n'est pas optimal.
   (b) Proposer un algorithme optimal pour résoudre le problème, et donner son coût.
3. On revient au cas où tous les livres ont la même hauteur $h = h_i, 1 \leq i \leq n$. On veut désormais ranger les n livres sur k étagères de même longueur L à minimiser, où k est un paramètre du problème. Il s'agit donc de partitionner les n livres en k tranches, de telle sorte que la largeur de la plus large des k tranches soit la plus petite possible.
   (a) Proposer un algorithme pour résoudre le problème, et donner son coût en fonction de n et k.
   (b) On suppose maintenant que la la taille d'un livre est en $2^{o(kn)}$. Trouver un algorithme plus rapide que le précédent pour répondre à la même question.

Exercice 8

Étant donné un tableau T de n entiers relatifs, on cherche $\max\{\forall i, j \in \{1 \ldots n\} : \sum_{k=i}^{j} T[k]\}$.
Par exemple pour le tableau suivant :

2	18	−22	**20**	**8**	**−6**	**10**	−24	13	3

L'algorithme retournerait la somme des éléments 4 à 7 soit 32.

1. Donner un algorithme retournant la somme maximale d'éléments contigus par une approche *diviser pour régner*.
2. Donner un algorithme retournant la somme maximale d'éléments contigus par *programmation dynamique*.
3. Comparer la complexité Asymptotique au pire cas des deux approches.
4. Peut-on adapter l'algorithme de programmation dynamique en dimension 2? Plus formellement, étant donnée une matrice M de $n \times m$ entiers relatifs, on cherche $\max\{\forall i, j \in \{1 \ldots n\}, \forall k, l \in \{1 \ldots m\} : \sum_{k_1=i}^{j} \sum_{k_2=k}^{l} M[k_1][k_2]\}$.

Formation Master 1 IDO
Optimisation Combinatoire 1
Examen session normale mars 2021

Exercice 2 (05 points)

1) **Définir les termes suivants :**
   - **La classe (de problèmes) P ;**
   - **La classe (de problèmes) NP ;**
   - **Une solution admissible d'un problème d'optimisation ;**
2) **A quoi sert la complexité d'un algorithme ?**
3) **Que signifie le qualificatif « combinatoire » d'un problème d'optimisation ?**

Exercice 2 (07 points)

Chacun des 2 algorithmes ci-contre permet de calculer la valeur d'un polynôme $p(x) = a_0+a_1x+a_2x^2+...+a_nx^n$ pour une valeur de x.
Le degré n est une donnée entière et les coefficients a_i sont stockés dans un tableau a[] de type float.

```
float P1 (int a[] , int n , float x)
{       s = a[0] ;
        for (i=1 ; i<=n ; i++)
                s+ = a[i]* x^i ;
        return s ;
}
```

```
float P2 (int a[] , int n , float x)
 {      s = a[0] ;
        q = 1;
        for (i=1 ; i<=n ; i++)
         {
                q =  q * x ;
                s +=  a[i]*q ;
         }
        return s ;
}
```

1. **Calculer le nombre de multiplications T1(n) que fait l'algorithme P1 en fonction de n puis exprimer sa complexité en notation O.**
2. **Calculer le nombre de multiplications T2(n) que fait l'algorithme P2 en fonction de n puis exprimer sa complexité en notation O.**
3. **Que peut-on déduire ?**
4. **Ecrire un algorithme P3 qui donne le même résultat en n-1 multiplications. Justifier.**

Exercice 3 (08 points)

Un examen de 4 heures est composé de 5 questions indivisibles, chaque question i a une note n_i et nécessite un temps de réponse t_i en heures. Ces détails sont récapitulés dans le tableau suivant :

Question i	1	2	3	4	5
Note n_i	10	6	12	9	8
Temps t_i	2	1	3	2	2

On désire déterminer, à l'aide d'un algorithme, quelles questions choisir pour avoir une note totale maximale.

1) **Il est clair que l'étudiant ne peut pas répondre à toutes les questions. Pourquoi ?**
2) **Quel est le problème modèle vu cours qui est applicable à celui-ci ?**

3) **Déterminer l'espace de recherche et calculer sa taille (nombre de solutions candidates).**
4) **Ecrire la formulation mathématique de ce problème d'optimisation.**
5) **Résoudre ce problème par la méthode de la programmation dynamique. (Donner la solution et la valeur de la solution).**
6) **Supposons que l'on dispose de n questions dans un examen de T heures, donner les complexités temporelle et spatiale de cet algorithme en fonction de n et T en notation O.**

Formation Master 1 IDO
Optimisation Combinatoire 1
Corrigé-type Examen session normale mars 2021

Exercice 1 (05 points)

1.

- **La classe P = ensemble de problèmes ayant des algorithmes polynomiaux déterministes pour les résoudre. ……..…1pt**
- **La classe NP = ensemble de problèmes ayant des algorithmes polynomiaux indéterministes pour les résoudre. ……..…1pt**
- **Une solution admissible = une solution vérifiant les contraintes du PO; ……..…1pt**

2. La complexité d'un algorithme sert à analyser, évaluer et comparer les algorithmes..1pt

3. « Combinatoire » signifie : l'espace de recherche est discret, fini et de taille assez importante. ……..…1pt

Exercice 2 (07 points)

1. T1(n)= 1+2+…+n=n(n+1)/2=O(n^2). ……..…1.5pt

2. T2(n)= 2+2…+2 = 2n =O(n). ……..…1.5pt

3. P2 est meilleur que P1. ……..…0.5pt

4.

```
float P3 (int a[] , int n , float x)
{   s = a[n] ;
    for (i=n ; i > 0 ; i--)
          s = s*x + a[i-1] ;
    return s ;
}
```

……..…2.5pt

En effet, la boucle for contient n itérations, chacune contient une multiplication. ……1pt

Exercice 3 (08 points)

1. Parce que $\sum_{i=1}^{5} t_i > 4\ heures$ ……..…0.5pt

2. Problème du sac-à-dos (KSP). ……..…0.5pt

3. Espace de recherche S= ensemble des parties de {1,2,3,4,5} et |S|= 2^5. ……..…1pt

4. $\begin{cases} max \sum_{i=1}^{i=5} x_i n_i \\ x_i \epsilon \{0,1\} \\ \sum_{i=1}^{5} x_i t_i \leq 4 \end{cases}$ **. ……..…1.5pts**

5. ……..…2.5pts

n t	0	1	2	3	4
0	0	0	0	0	0
1	0	0	10	10	10
2	0	6	10	16	16
3	0	6	10	16	18
4	0	6	10	16	19
5	0	6	10	16	19

La solution optimale est {1,4} et f{1,4}= n1+n4=19……..…1pt

1. Complexité temporelle = Complexité spatiale = O(n.T) ……..…1pt

Formation Master 2 IDO
Module Méthodes d'optimisation
Examen session normale février 2021

Exercice 1 (08 points)

Une école spécialisée en informatique offre, périodiquement à ses visiteurs, des formations dans 5 modules, chaque module est caractérisé par son nombre de crédits et son volume horaire. Les modules d'une période donnée sont détaillés dans le tableau suivant :

Module i	1	2	3	4	5
Crédit ci	8	5	2	7	4
VH hi	24	20	30	40	12

Pour chaque visiteur, l'école octroie une formation gratuite de 50 heures couronnée par une attestation mentionnant le nombre de crédits acquis. On se propose de déterminer quels modules choisir par le visiteur afin de totaliser un crédit maximal (les modules ne doivent pas être fractionnés).

1) **Quel est le type de ce problème d'optimisation ? quel est le modèle vu cours qui lui est applicable ?**
2) **Déterminer l'espace de recherche et calculer sa taille.**
3) **Ecrire la formulation mathématique de ce problème d'optimisation.**
4) **On se propose de résoudre ce problème par un algorithme de colonie de fourmis.**

a) **Quelles sont les composantes d'une solution.**

b) **On définit la distance entre deux composantes i et j d'une solution x par d(i,j)=|hi/ci – hj/cj|+1.**

A la première itération, une fourmi ayant déjà choisi le module 1, quelles sont les probabilités pour qu'elle choisisse les modules 2,3,4,5 respectivement juste après. (On prend $\alpha = \beta = \tau 0 = 2$.).

c) Quelle quantité de phéromone serait ajoutée à l'arrête 1-2 si cette fourmi prenait le module 2.

Exercice 2 (12 points)

L'algorithme A ci-contre permet de résoudre un problème d'optimisation combinatoire.

```
void A() {
float F(float[] x) { return abs(x[0]*x[0]+ x[1]*x[1]- x[2]*x[2]) ;}
g[] = [1,1,1] ;
for ( i=0; i<m ; i++)
   for ( j=0; j<3 ; j++)
      {  x[i] [j] = 1+int(100*random(0,1)) ;
         v[i] [j] = int(4*random(0,1)) ;
         p[i] [j ] = x[i] [j] ;
         if (F(p[i]) < F(g))  g[] = p[i] ;
      }
for (t =0; i<k ; t++)
  for ( i=0; i<m ; i++)
    for ( j=0; j<3 ; j++)
    { v[i][j]=0.8*v[i][j]+0.5*abs(x[i][j]- p[i][j])+0.5*abs(x[i][j]- g[j]) ;
       if ( v[i][j] > 4) v[i] [j]= 4 ;
       x[i][j]= x[i][j] +  v[i][j] ;
       if ( x[i] [j] >100) x[i] [j]=100 ;
       if (F(x[i]) < F(p[i]) )
            { p[i] = x[i];
              if (F(p[i]) < F(g))   g[] = p[i] ;
            }
      }
  s.o.p (g[] , F(g[]))
}
```

5. **En relever la formulation mathématique de ce problème, l'espace de recherche et sa taille.**
6. **Que fait exactement l'algorithme A ?**
7. **Qu'appelle-t-on cette méthode et quels sont ses paramètres (noms et valeurs).**
8. **Evaluer la complexité de cet algorithme.**
9. **On se propose de résoudre ce même problème par un algorithme génétique avec une population de 4 individus en utilisant la sélection par roulette. a) Proposer une fonction fitness puis compléter le tableau suivant :**

Individu	Fitness	Probabilité de sélection
(1,1,1)		
(2,3,4)		
(5,5,7)		
(6,8,9)		

b) Implémenter un opérateur de mutation de votre choix.

Formation Master 2 IDO

Module Méthodes d'optimisation

Corrigé-type Examen session normale février 2021

Exercice 1 (08 points)

1) **Discret. KSP.** (0.5+0.5 pt)
2) **Espace de recherche = ensemble des parties de {1,2,3,4,5} ; | Espace de recherche| = $2^5 = 32$.** (0.5+0.5 pt)
3) **Formulation mathématique :** $\begin{cases} max \ \sum_{i=1}^{5} x_i c_i \\ x_i \in \{0,1,\} \\ \sum_{i=1}^{5} x_i h_i \leq H \end{cases}$ (1.5 pt)

4.a) Les composantes d'une solution = les modules (0.5 pt)

4.b) (3 pt)

Module	Proba
2	$\frac{(\tau_{12})^{\alpha}(\eta_{12})^{\beta}}{(\tau_{12})^{\alpha}(\eta_{12})^{\beta}+(\tau_{15})^{\alpha}(\eta_{15})^{\beta}} = \frac{2^2(\frac{1}{\left\lvert\frac{h_1}{c_1}-\frac{h_2}{c_2}\right\rvert+1})^2}{(\tau_{12})^{\alpha}(\eta_{12})^{\beta}+(\tau_{15})^{\alpha}(\eta_{15})^{\beta}} = \frac{1}{5}$
3	0 contrainte non satisfaite (pas d'arrête 1-3)
4	0 contrainte non satisfaite (pas d'arrête 1-4)
5	$\frac{(\tau_{15})^{\alpha}(\eta_{15})^{\beta}}{(\tau_{12})^{\alpha}(\eta_{12})^{\beta}+(\tau_{15})^{\alpha}(\eta_{15})^{\beta}} = \frac{2^2(\frac{1}{\left\lvert\frac{h_1}{c_1}-\frac{h_5}{c_5}\right\rvert+1})^2}{(\tau_{12})^{\alpha}(\eta_{12})^{\beta}+(\tau_{15})^{\alpha}(\eta_{15})^{\beta}} = \frac{4}{5}$

c) quantité de phéromone ajoutée à l'arrête 1-2 $= \frac{1}{d(1,2)} = 1$ (1 pt)

Exercice 2 (12 points)

1) **La formulation mathématique :** $\begin{cases} min \ (x_0{}^2 + x_1{}^2 - x_2{}^2) \\ x_i \in N \\ 0 \leq x_i \leq 100 \end{cases}$ (1.5 pt)

Espace de recherche = $[1,100]^3$; | Espace de recherche| = 10^6. (0.5+0.5 pt)

2) **Cet algorithme recherche les entiers x,y,z compris entre 1 et 100 vérifiant $x^2+y^2=z^2$.** (1 pt)
3) **PSO** (0.5 pt)

Paramètres : (1.5 pt)
Taille de la population = m
Nombre d'itérations = k
Facteur d'inertie = 0.8
Facteurs d'accélération = 0.5 et 0.5
Vitesse maximale = 4

1) **Complexité : O(3m)+ O(3mk) = O(3mk)** (1.5 pt)
a) **Fonction fitness : f(x,y,z)= 1/(abs($x^2+y^2 - z^2$) +1)** (1 pt)

Individu	Fitness	Probabilité de sélection
(1,1,1)	1/2	5/13
(2,3,4)	1/4	5/26
(5,5,7)	1/2	5/13
(6,8,9)	1/20	1/26
	13/10	1

2.5

5.b) Mutation

Spécification : on ajoute 1 à l'une des composantes, si on dépasse 100, on retranche 1. (0.5 pt)

```
Mutaion (int x[])
{  ind = int(3*random(0,1)) ;
   x[ind] = x[ind] +1 ;
   if (x[ind] > 100 )
       x[ind] =  x[ind] -2 ;  } (1 pt)
```