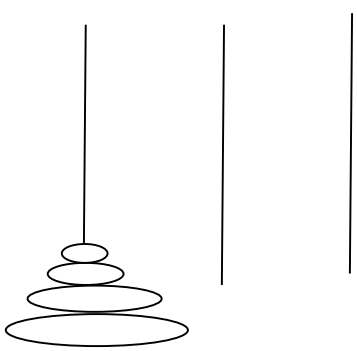
USTHB, FEI,
Département d'Informatique
Master SII, S2

Bab-Ezzouar le 27 Mai 2017

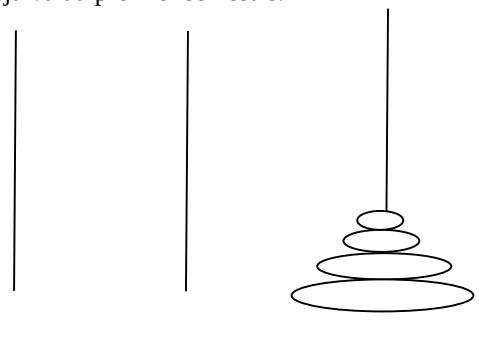
EMD sur les méta-heuristiques

Background : maitrise de l'intelligence en essaim pour la résolution de problèmes.

Considérer le problème de la tour de Hanoi (PTH en abrégé) déjà vu au premier semestre.



Configuration initiale de la tour de Hanoi



Configuration finale de la tour de Hanoi

Le problème est un jeu défini formellement comme suit :
Instance : Il existe 3 piquets ou tiges A, B et C sur lesquels n disques sont déposés. La configuration initiale de PTH fait apparaître les n disques sur le piquet A du plus grand au plus petit et aucun sur les deux autres tiges. Dans la configuration cible, les n disques apparaissent sur le piquet C (figure ci-dessus). Le jeu consiste à déplacer la pile des n disques se trouvant sur le piquet A de la configuration initiale vers le piquet C en respectant les contraintes suivantes :

- ne déplacer qu'un seul disque à la fois
- ne pas poser un disque sur un autre de plus petite taille
- se servir du piquet B pour les déplacements intermédiaires

Question : Existe-il une série de déplacements commençant par la configuration initiale et se terminant par la configuration finale qui respecte les contraintes énoncés dans la description de l'instance ?

1) Décrire l'approche espace des solutions pour PTH et définir en particulier : (1 pt)

L'approche espace des solutions pour PTH repose sur la recherche d'une solution de grande qualité dans l'espace total des solutions en utilisant un algorithme évolutionnaire.

- Une solution de PTH. Quelle est la difficulté rencontrée ? (1 pt)

Une solution pour PTH est une séquence de déplacements d'un seul disque à la fois commençant par l'état initial tout en respectant la contrainte énoncée dans la définition du problème.
Les déplacements possibles sont codés comme suit :
1 : de A vers B
2 : de A vers C
3 : de B vers A
4 : de B vers C
5 : de C vers A
6 : de C vers B
Nous aurons besoin de la notion de successeur pour l'implémentation des algorithmes évolutionnaires. Les successeurs des déplacements sont comme suit :
successeur (1) := {3,4}
successeur (2) := {5,6}
successeur (3) := {1,2}
successeur (4) := {5,6}
successeur (5) := {1,2}
successeur (6) := {3,4}

La difficulté rencontrée est la détermination de la longueur de la solution qui est inconnue au lancement du processus de recherche.

Comment y remédier ? (1 pt)

Définir une longueur assez importante au début puis la fixer par des expérimentations.
Si la longueur est égale à L=6 par exemple, une solution serait :
x = 2 5 1 3 1 4
La déclaration du type de la solution est :
Var x : tableau[1..L] of 1..L ;

- L'espace de recherche. (1 pt)

C'est l'ensemble de toutes les solutions potentielles au problème. Sa taille est également inconnue car la longueur de la solution n'est inconnue qu'à la fin du processus de recherche

- Un algorithme évolutionnaire. (1 pt)

Un algorithme évolutionnaire consiste à rechercher dans l'espace des solutions la solution optimale ou à défaut une solution de grande qualité proche de la solution optimale. Il initialise le processus de recherche avec une ou plusieurs solutions tirées aléatoirement ou par le biais d'une heuristique puis fait évoluer ces solutions vers d'autres solutions de meilleure qualité. Il s'arrête dans l'une des trois conditions suivantes :

- lorsqu'il atteint la solution optimale ou bien
- lorsque les ressources calculatoires sont épuisées ou bien
- lorsque la solution stagne sans amélioration de sa qualité.

2) Parmi les algorithmes BSO, ACS et PSO vus en cours, lequel est le plus adéquat pour résoudre PTH ? (1 pt) Pourquoi ? (1 pt)

Pour faire évoluer une solution, BSO utilise deux stratégies :

- La détermination de solutions distantes de la solution de référence
- le concept de voisinage pour l'exploration de la région d'une solution de la zone de recherche.

Ces deux opérations peuvent construire des solutions non réalisables car le changement d'un déplacement de la séquence des déplacements de la solution peut perturber l'enchainement cohérent des déplacements. BSO peut être alors implémentée difficilement pour PTH car à chaque détermination d'une nouvelle solution, il faut corriger la solution si elle n'est pas réalisable ou la rejeter. Cette tâche nécessite un temps de calcul énorme.
Pour PSO, le calcul de nouvelles solutions se fait différemment en utilisant les règles de transition. Cependant comme ces règles utilisent le calcul des distances, les solutions nouvelles peuvent ne pas être valides. PSO aura aussi des difficultés à être implémenté.
Pour ACS, comme la fourmi artificielle construit une solution composant par composant, elle prendra la précaution de ne considérer que les composants fiables à chaque étape pour construire une solution valide.
En conclusion c'est ACS qui convient le mieux pour PTH.

3) Donner l'ossature générale de l'algorithme PSO pour PTH. (1 pt)

```
begin
  Initialize N particles: positions and velocities;
  Evaluate the particles positions;
  for each particle i do Pbest_i = x_i;
  calculate Gbest using the transition rule;
  for each iteration do
   for each particle p do
      update the velocity and the position;
      if the position is valid then
            move the particle and evaluate its fitness;
            update Pbest;
      endif;
    endfor;
    update Gbest;
  endfor;
end
```

Initialization

For each particle do
- ✓ Set its position at random; (* build randomly a valid solution of length l *)
- ✓ Draw its velocity at random; (* draw at random an integer value for the velocity *)
- ✓ Set its position to $\boldsymbol{Pbest_i}$ (* $\boldsymbol{Pbest_i}$ = solution *)

Set *Gbest* as the best solution $Pbest_i$ (* ***Gbest*** = best of $\boldsymbol{Pbest_i}$ *)

Evaluate the particle fitness:
Compute its fitness : *f(x)= heuristic function of the last component of the solution.*

```
function valid (x:solution) : Boolean;
var     i: 1..L;
        accept, constraint_violation: Boolean;

begin
        accept := true;
        constraint_violation := false;
        i := 1
        while  (i < L) and valid do
          if successeur (x[i-1]) does not contain x[i]
                then accept := false;
          if the disk of the start of the move is larger than the disk of the end of the move
                then constraint_violation := true;
        return (accept and not constraint_violation);
end
```

- ➢ Comment se fait l'interaction entre les particules ? (2 pts)

L'interaction se traduit par le fait qu'une particule tient compte de la meilleure position du groupe pour se déplacer conformément à la règle de transition.

- ➢ Décrire les règles de transition d'une particule. (2 pts) Que pouvez-vous conclure ? (1 pt)

Updating the velocity:
$v(t+1) =$ partie entière $(w*v(t) + c_1 r_1 *(Pbest - x(t)) + c_2 r_2 *(Gbest - x(t)))$
where:
- x(t) the current position of the particle.
- *Pbest* is the best visited position of the particle.
- *Gbest* is the best position found by the swarm.
- w, c_1, c_2 are empirical parameters.
- r_1, r_2 are random numbers.
-

Updating the position:
$x(t+1) = x(t) + v(t+1)$
Cette opération consiste à modifier v(t+1) composants de x de manière aléatoire. Une idée serait de changer les déplacements qui n'enchainent pas correctement avec les déplacements précédents. C'est une manière de corriger et de rendre valide une solution engendrée par la règle de transition de la particule.

4) Donner l'ossature générale de l'algorithme ACS pour PTH. (1 pt)

```
begin
  Pheromone and parameters initialization ();
  for i=1 to Max-Iter do
  begin
    for k=1 to Nb-ants do
     begin
      build a solution (sk);
      evaluate (sk);
      apply online delayed update of pheromone();
     end;
     determine the best solution of the iteration();
     apply offline delayed update of pheromone();
```

end;
end.

- Comment se fait l'interaction entre les fourmis ? (2 pts)

L'interaction entre les fourmis se fait grâce à la phéromone sécrétée par les congénères.

- Expliciter les fonctions utilisées dans l'algorithme. (1 pts)

Procedure Build (var s)**;**
Input: D={1,2, …, L} the set of all moves;
Output: s;
begin
s : = empty;
s[1] := a random number from D;
i := 2;
while (*i* < *L*) **do**
select *s[i]* from ***successeur(s[i-1])*** *using the transition rule*;
i := i+1;
endwhile;
end.

- Donner et expliquer clairement les règles de transition d'une fourmi. Que pouvez-vous conclure ? (1 pt)

Pseudo random proportional rule:

if $q <= q_0$
then

$$P_i^k(t) = \begin{cases} 1 & \text{if i} = \text{argmax}\{[T_i(t)]^\alpha [\eta_i]^\beta\} \\ 0 & \text{otherwise} \end{cases}$$

else

$$P_i^k(t) = \frac{[T_i(t)]^\alpha [\eta_i]^\beta}{\sum_{l \in L} [T_l(t)]^\alpha [\eta_l]^\beta}$$

where l := successeur(s[i]), T_i is the amount of pheromone on s[i] and
η_i = heuristic evaluation of s[i].

- Donner et expliquer clairement les règles de mise à jour de la phéromone. (2 pts)

Online update step by step of the pheromone **us**ing the following rule :

$T_i(t+1) = (1-\rho)\, T_i(t) + \rho\, \tau_0$ [0<ρ<=1]

τ_0 is the initial value of the pheromone, which is small.

ρ is the alteration parameter.

Offline update using the rule :

$T_i(t+1) = (1-\rho)\, T_i(t) + \rho\, \Delta T_i^{best}(t)$

$\Delta T_{ij}^{best}(t)$ is the added pheromone quantity, which is proportional to the **best solution** quality found at the current iteration.

$\boldsymbol{\Delta T_i^{best}(t) = T_i^{best}(t+1) - T_i^{best}(t)}$

FIN !