Rapport : Methodes et outils de l'IA et la RO

mini-projet

Becirspahic Lucas

Université Pierre et Marie Curie - 13 mars 2017

Introduction

1 Implémentation de strategies

2 La strategie contre

Dans cette strategie on suppose que l'on connait la strategie de l'adversaire. Soit un chemin la liste ordonée des fioles par lequel un personnage va passer, on le représente comme une liste de tuple (fiole, distance).
L'attribut distance représente la distance commulé pour aller à cette fiole en passant pas les états antérieurs de mon chemin.

Soit $s1$, le chemin de mon joueurs en se basant sur la strategie qui prend la meilleur fiole possible dans une distance raisonnable.
Soit $s2$, le chemin de mon adversaire optenu en appliquant sa strategie.
Soit la fonction $distance(chemin, fiole)$ la distance qu'il me faut pour atteindre la fiole en passant par le chemin

On définit un dictionnaire de gain nomé $dicoGain$ pour chaque fiole de la manière suivante :
– *Si distance(s1,fiole) > distance(s2,fiole) alors dicoGain[fiole] = valeur de la fiole*
– *Sinon dicoGain[fiole] = -1 * valeur de la fiole*

Une fois ses choses définit, on arrive au coeur de l'algorithme, on cherche à maximiser :

$$gain = \sum_{f \in fioles} dicoGain[f]$$

Pour ce faire, on dispose d'une opération de permutation qui change l'ordre de $s1$ en fonction de $s2$. On définit cette opération de la manière suivante :

Algorithme : Permutation(chemin du joueur,chemin de l'adversaire,fiole,case)
Entrées : $s1$: chemin, f : fiole, $case$: case
$old \leftarrow$ chemin du point de départ jusqu'a case -1
$new \leftarrow$ on recalcule le chemin et les distances pour les fioles restantes
$chemin \leftarrow$ on concatene old et new
retourner $chemin$

Puis, on souhaite trouver la meilleur permutations parmis celle possible de la manière suivante :

Algorithme : MeilleurPermutation(chemin du joueur, chemin de l'adversaire, fioles)

Entrées : $s1$: chemin, $s2$: chemin de l'adversaire, $fioles$: liste de fioles

pour $f \in fioles$ **faire**
  pour $case \in s1$ **faire**
    $new = Permutation(s1, f2, f, case)$
    On calcule le dicoGain pour le nouveau chemin
    new gain ← *la nouvelle valeur du gain*
    si $newgain > gain$ **alors**
      $chemin \leftarrow new$
      $gain \leftarrow new\ gain$
retourner $chemin$

Pour pouvoir estimer la cohérence de notre probabilité empirique, calculons une approximation de la probabilité qu'un mot w apparaisse n fois dans une séquence aléatoire à l'aide de lois usuelles.

Soit P_{th} la probabilité d'avoir un mot w, de longueur k, qui apparaît n fois dans une séquence aléatoire de longueur l, avec $0 \leq n \leq (l-k+1)$. Soit N la variable aléatoire qui compte le nombre de fois où le mot apparaît dans une séquence. On suppose sur les positions d'occurrences d'un mot sont indépendantes. Notons qu'on peut calculer P_{th} par la formule suivante, quelle que soit l'approche abordée :

$$P_{th}(N \geq n) = 1 - P_{th}(N < n) = 1 - \sum_{i=0}^{n-1} P_{th}(N = i)$$

– Première approche
Pour les $(l-k+1)$ positions différentes de la séquence (les mots ne se chevauchent pas), il y a un seul choix de mot sur tous les mots possibles. Or, il existe 4 bases nucléiques possibles (`A, C, G` ou `T`) pour chaque lettre du mot de longueur k, il y a donc 4^k mots possibles. Ainsi, N suit une loi binomiale :

$$N \sim \mathcal{B}(l-k+1, \frac{1}{4^k})$$

Posons $q = l-k+1$. On obtient alors :

$$P_{th}(N = n) = C_q^n \left(\frac{1}{4^k}\right)^n \left(1 - \frac{1}{4^k}\right)^{q-n}$$

On se dote d'une fonction simple, qui calcule cette probabilité selon la formule ci-dessus.

```
def proba_theorique(k, n, l):
    q = l - k + 1
    a = binom(q, n)
    b = (1.0/4**k)**n
    c = (1 - (1.0/4**k))**(q-n)
    return a*b*c
```

Calculons la probabilité théorique qu'un mot de longueur 6 (par exemple `ATCTGC`) soit présent au moins une fois dans la séquence PHO. On trouve ainsi $P_{th}(N \geq 1) = 1 - P_{th}(N = 0) \approx 0.629$. Or, la valeur empirique de cette probabilité est environ 0.557, ce qui n'est pas cohérent. En effet, dans cette approche, on ne prend pas en compte les fréquences de chaque lettre Par exemple, le mot `ATCTGC` a -en principe- plus de chance d'apparaître que le mot `AAAAAA`, car il possède plus de lettres distinctes.

– Seconde approche
Soit $w = w_1 w_2 ... w_k$ le mot dont on veut déterminer le nombre d'apparitions. Soit $\mathcal{F}(w_i)$ la fréquence de la

lettre w_i dans une séquence. La probabilité que w apparaisse est le produit des probabilités de chacune de ses lettres. La variable aléatoire N suit alors une loi binomiale dont on a modifié un paramètre :

$$N \sim \mathscr{B}(l-k+1, \prod_{i=1}^{k} \mathscr{F}(w_i))$$

Ainsi, en posant comme précédemment $q = l-k+1$, on obtient :

$$P_{th}(N=n) = C_q^n \left(\prod_{i=1}^{k} \mathscr{F}(w_i) \right)^n \left(1 - \prod_{i=1}^{k} \mathscr{F}(w_i) \right)^{q-n}$$

En calculant la probabilité que le mot `ATCTGC` apparaisse au moins une fois, étant donné le quadruplet des fréquences des lettres dans la séquence PHO, on trouve $P_{th}(N \geq 1) \approx 0.542$, pour une probabilité empirique de 0.557. Cette approche est donc plus précise que la précédente.