HMIN233
Algorithmique d'exploration et de mouvement

Rapport TP4 et TP5

Auteur :

Canta Thomas
Fontaine Quentin

Master 1 - AIGLE/DECOL
Faculté des sciences de Montpellier
Année universitaire 2020/2021

Table des matières

TP4 - Algorithme Génétique

Résultat

Pour chaque résultat qui suit la méthode brute et l'algorithme génétique obtenez le même.

Taille population	Nb cycle	Best selection	Nb villes	Brutes	Algo Gen
1000	3000	30%	10	99ms	847ms
1000	7500	30%	12	8126ms	2134ms
1000	7000	40%	12	8132ms	1934ms

Dans la majorité des cas, nous avons approximativement 1 chance sur deux d'obtenir le meilleur résultat, sinon on s'en approche de près à $\pm$ 75 mètres linéaires.

Notre voyageur de commerce obtient systématique le même résultat que la méthode brute. Néanmoins, quand la population et le nombre de cycles est trop faible la méthode brute trouve un bien meilleur résultat. Par exemple, pour une population de *1000 individus*, *3000 cycles* et en prenant à chaque mutation *les 30% des meilleurs*, nous avons obtenu un score de *1481* pour la méthode brute contre *1789* avec voyageur de commerce.

Optimisation

Ayant utilisé en début de semestre la métaheuristique du recuit simulé (pour MAX-CSP) nous nous demandions si celle-ci permettrait d'optimiser notre cas, et nous avions raison. D'après une recherche menée, cela fonctionnerait d'autant plus si l'on ajoute l'algorithme 2-opt dans l'équation, permettant de réduire un temps de calcul de $\approx$ 30h à $\approx$ 38min [1].

Il existe aussi différentes approches telles que l'hypothèse de l'inégalité triangulaire (dans un cas métrique). Un graphe respectant l'inégalité triangulaire est un graphe ou pour tout point, prendre l'arête entre deux sommets est toujours moins coûteux que de passer par des sommets intermédiaires (cf : voir exemple page 6, Figure 3). Mais ceci ne fonctionne que si les poids du graphe représentent une mesure (d'où le cas métrique). Cette approche permettrait donc de réduire un bon nombre de recherches inutiles, donc de réduire considérablement le temps de calcul (l'algorithme de Christofides, connue pour être l'un des plus rapides pour cette approche) [2].

1. source : http://tchiling.iiens.net/Documents/TIPE/Dossier.pdf (page 12 et 13)
2. source : https://www.youtube.com/watch?v=yqH11OHfN2U

TP5 - Neurocontroleurs

Résultat

Go To Object

Sans apprentissage notre agent tourne en rond mais après quelque minutes il devient capable de se repositionner afin d'atteindre le point d'arrivée (voir résultat ci-dessous).

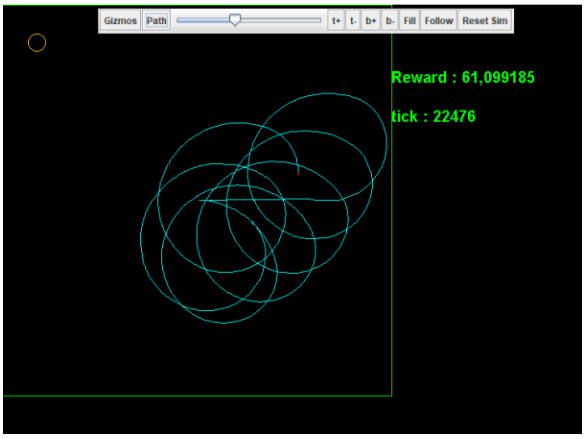

FIGURE 1 – Test sans apprentissage

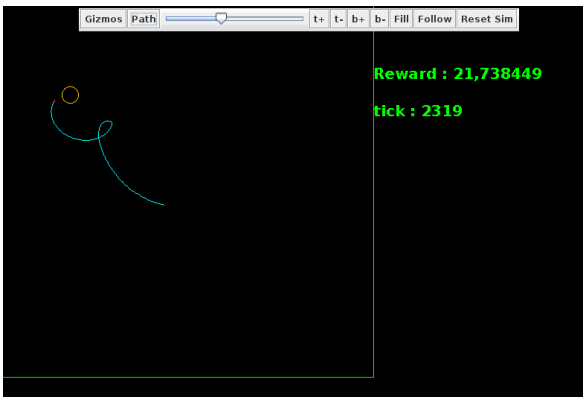

FIGURE 2 – Test avec apprentissage

Le temps d'apprentissage est quand même assez long pour obtenir un tel résultat, ici ≈4 minutes ce sont écoulées entre les deux images.

AVOID

Pour le makeMind(), nous avons :

- L'actuator "MotR",
- L'actuator "MotL",
- Les capteurs "S1, S2, S3, S9 et S10", qui sont les capteurs devant l'agent.

Pour cette partie il nous semble judicieux de réutiliser les fonctions de contrôle et de récompenses de GTO (sauf RW_SensorOverThreshold) en y ajoutant celle-ci :

- RW_AvoidObs ;
  Si l'agent évite un obstacle, alors on le récompense.
- RW_ClosestObstaclePenalty ;
  Si l'agent s'approche trop près d'un obstacle, alors on lui retire des points.
- RW_PunishOnCollision ;

Si l'agent touche obstacle, alors on lui retire des points.

- RW_StraightMovement
  On 'force' l'agent à essayer d'aller tout droit afin d'éviter qu'il tourne sur lui-même.

NB : Nous avons essayés d'utiliser la récompense RW_AvoidObs mais celle-ci faisait tourner notre agent, très régulièrement, sur lui-même tel un poisson rouge.

GTOA

Il suffit de d'utiliser à la fois les fonctions et contrôle de GOT et Avoid.

Collect

Pour le makeMind(), nous avons :

- L'actuator "MotR",
- L'actuator "MotL",
- Les capteurs "S1, S2, S3, S9 et S10", qui sont les capteurs devant l'agent,
- Le capteur "DISTOBJ",
- Le capteur "RADOBJ",
- Le capteur "SENSOBJ",
- Le capteur "DISTDZ",
- Le capteur "RADDZ",
- Le capteur "SENSDZ",

Nous n'avons pas implémenté cette partie mais nous pouvons évoquer nos suppositions. Nous pouvons réutiliser, les contrôles et récompenses de GTO dans un monde sans obstacles, GTOA sinon. On ajoute à cela :

- RW_Collect
  Si l'agent est à proximité de la cible il l'a collecte.
- RW_Explorer
  Si l'agent visite des lieux "non visités" alors on le récompense.
- RW_BelowDistanceFromTarget
  Si l'agent se rapproche de ses cibles, on le récompense.

Annexes

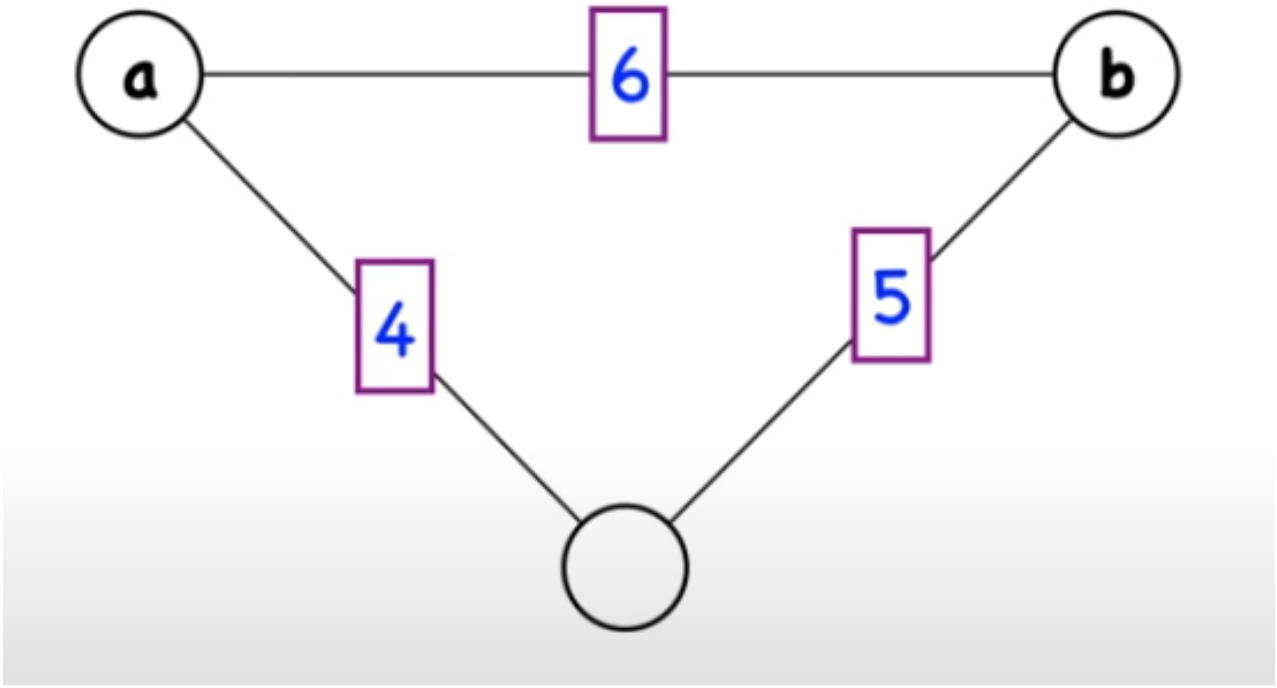

FIGURE 3 – Exemple d'un graphe respectant l'inégalité triangulaire