Université de Montpellier / Faculté Des Sciences

Recherche de chemin

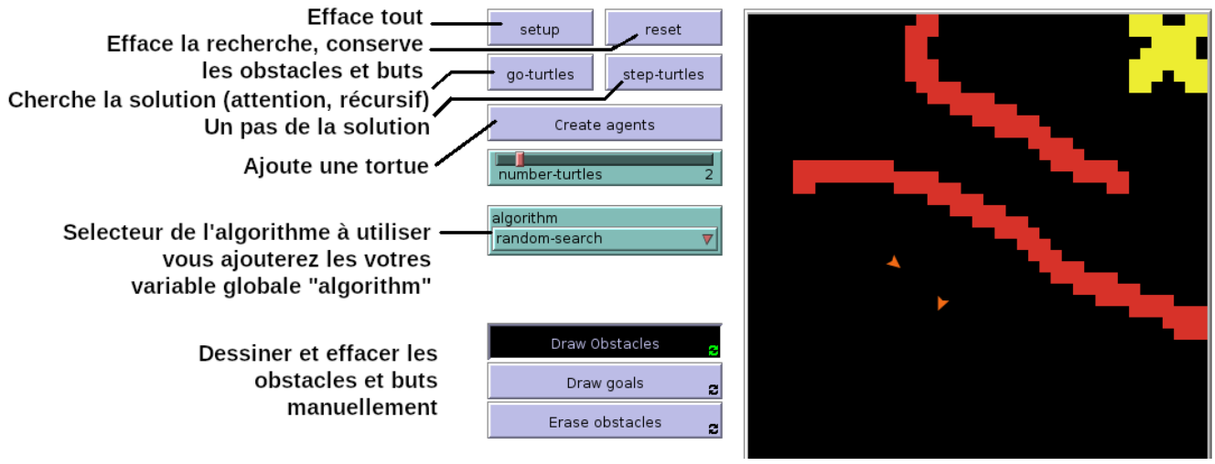
Algorithmes de mouvement — HMIN 233

Suro François
(adaptation des cours de Jacques Ferber)

8 décembre 2020

Dans ce TP nous implémenterons différents algorithmes de recherche de chemins dans NetLogo. Vous vous baserez sur le projet `search-algo-template-minimal.nlogo` qui contient un squelette de l'algorithme de recherche et des fonctions pour placer des buts et obstacles.

Voici un court descriptif de l'interface :



Vous répondrez aux questions en modifiant la section "Algorithme de recherche" du code. En l'état, ce programme effectue une recherche aléatoire.

1 Algorithmes de base, sans information

Implémentez un algorithme d'exploration en utilisant d'abord l'algorithme en profondeur/largeur d'abord. On rappelle l'algorithme général de base :

```
to Search
   list-nodes <- insert(initial-node)
   Tant que pas trouve
      Si list-nodes est vide, alors echec ;; on n'a pas trouve de solution.
      cnode <- first (sort(list-nodes))   ;; trier et extraire cnode de la liste.
      evaluer(cnode) ;; Executer ce qu'il y a a faire, evaluer le noeud.
                     ;; si on trouve le but directement, sortir.
                     ;; et retourner la solution.
      list-nodes <- add-list (generate (cnode), list-nodes)
      ;; en profondeur d'abord si les noeuds sont places devant.
      ;; en largeur d'abord si les noeuds sont places apres.
]
```

Pour l'instant, un état (un nœud de l'exploration) est donné par : [patch, valeur/coût, parent]
Si besoin, vous modifierez les procédures suivantes :

- search-step : effectue une étape de la recherche.
- sort-nodes : tri les nœuds si nécessaire.
- merge-nodes : fusionne la liste produite par generate avec la liste des nœuds.

- generate : qui crée une liste de nœud à explorer à partir du nœud courant.

Vous ajouterez aussi un moyen d'afficher le chemin trouvé par l'algorithme de recherche, en colorant les patches.

Testez ces algorithmes avec différentes configurations de l'environnement (obstacle , buts ...)

2 Algorithme avec information : Dijkstra et A*

Implémentez maintenant Dijkstra et A*. Ajoutez ces deux algorithmes à votre sélecteur afin de pouvoir passer facilement de l'un à l'autre

- **Dijkstra** utilise une fonction de coût qui porte uniquement sur le chemin effectué.
- **A*** utilise deux fonctions : l'estimation h(n) et le coût g(n), pour calculer une valeur d'un noeud : f(n) = g(n) + h(n). Le coût est donné par le nombre de mouvements nécessaires pour venir jusqu'ici, et l'estimation comme une évaluation du nombre de coups qu'il serait nécessaire pour aller directement au but.

  La fonction h est une heuristique (d'où son nom). Par défaut on peut prendre une distance (distance euclidienne, ou distance de Manhattan).

Il vous faudra sûrement modifier la structure du nœud d'exploration pour ajouter ces informations.

Comparez la résolution du problème par les différents algorithmes, sur différents problèmes. Cherchez des cas particuliers.