DUMONT laurent

MA60-Informatique (HD)

20170834

IHDC – M036 – Techniques d’Intelligence Artificielle

-

Devoir 1 – Recherche dans un espace d’états

Université de Namur – Année académique 2018-2019

Intuition

La solution imaginée s’inspire du problème des maris jaloux. En tenant compte des éléments suivants :

* Les contraintes sur la présence des personnes disparaissent.
* La torche joue le même rôle que la barque
* Le temps doit être calculé,

Nous sommes face au même type de problème à résoudre.

Il est donc possible de s’inspirer de la solution du problème des maris jaloux pour résoudre le problème de la traversée d’un pont avec une torche.

Etat

Un état peut être décrit par la liste constituée des éléments suivants :

* Gauche : une liste qui mentionne quels sont les membres du groupe U2 présents sur la rive gauche.
* Droite : une liste qui mentionne quels sont les membres du groupe U2 présents sur la rive droite.
* Torche : une liste qui mentionne le ou les membres du groupe qui ont la torche.
* PosTorche : indique si la torche est sur la rive gauche ou droite
* Horloge : mentionne le temps en cours utilisé pour traverser le pont.

EE = {(Gauche, Droite, Torche, PosTorche, Horloge) :

Gauche = [Gi] où 0 ≤ i≤4,

Droite = [Di] où 0 ≤ j≤4,

Torche = [Ti] où 0 ≤ k≤2,

i + j + k = 4,

PosTorche = {gauche, droite},

Duree(Bono) = 1, Duree(Edge) = 2,

Duree(Adam) = 5, Duree(larry) = 10,

Horloge = Horloge + max(Duree(Torche[Uj]))

}

Etat initial

{[bono, edge, adam, larry],[],[],gauche,0}

Etat(s) final(aux)

Succès : {[],[bono, edge, adam, larry] ,[],droite,0..17}

Echecs : {[],[bono, edge, adam, larry] ,[],droite,17..∞}

Opérateurs

* partir : EE x EE x N x N -> EE
* traverser : EE -> EE
* arriver : EE x EE x N x N-> EE

partir(([Gw,Gx,Gy,Gz],\_ ,[],gauche,\_),i,j)

=

([Gw,Gx],\_,[Ti,Tj],gauche,\_)

Si

{

y = i et z = j

}

partir(([Gw,Gx,Gy,Gz],\_ ,[],gauche,\_),i,\_)

=

([Gw,Gx,Gy],\_,[Ti],gauche,\_)

Si

{

z = i

}

partir(([Gw,Gx,Gy,Gz],\_ ,[],droite,\_),i,j)

=

([Gw,Gx],\_,[Ti,Tj],droite,\_)

Si

{

y = i et z = j

}

partir(([Gw,Gx,Gy,Gz],\_ ,[],droite,\_),i,\_)

=

([Gw,Gx,Gy],\_,[Ti],droite,\_)

Si

{

z = i

}

traverser([Gw,Gx,Gy],\_,[Ti],gauche,\_)

=

[Gw,Gx,Gy],\_,[Ti],droite,\_)

traverser([Gw,Gx],\_,[Ti,Tj],gauche,\_)

=

[Gw,Gx],\_,[Ti,Tj],droite,\_)

traverser([Gw,Gx,Gy],\_,[Ti],droite,\_)

=

[Gw,Gx,Gy],\_,[Ti],gauche,\_)

traverser([Gw,Gx],\_,[Ti,Tj],droite,\_)

=

[Gw,Gx],\_,[Ti,Tj],gauche,\_)

arriver(\_,\_,[Ti,Tj],gauche,Horloge1)

=

([Gw,Gx],\_,[],gauche,Horloge2)

Si

{

i = w et j = x,

Horloge2 = Horloge1 + max(Durée(Ti, Tj))

}

arriver(\_,\_,[Ti],gauche,Horloge1)

=

([Gw],\_,[],gauche,Horloge2)

Si

{

i = w,

Horloge2 = Horloge1 + Durée(Ti)

}

arriver(\_,\_,[Ti,Tj],droite,Horloge1)

=

(\_,[Gw,Gx],[],droite,Horloge2)

Si

{

i = w et j = x,

Horloge2 = Horloge1 + max(Durée(Ti, Tj))

}

arriver(\_,\_,[Ti],droite,Horloge1)

=

(\_,[Gw],[],droite,Horloge2)

Si

{

i = w,

Horloge2 = Horloge1 + Durée(Ti)

}