Examen mi-session

Intelligence Artificielle II (IFT-17587)

Jeudi 21 février 2001

De 15h30 à 18h20 en salle PLT-2341

- *Les documents permis sont le livre, les acétates du cours et une double page de notes.*
- *Les temps entre parenthèses représentes les temps suggérer pour répondre à chacune des questions.*
- *Le nombre de points accordés à chacune des questions est aussi inscrit entre parenthèses.*

---

1. (15 pts) (20 min) Pour chacun des exemples d'agents suivants, dites quelle architecture d'agent est la plus appropriée (simple réflexe, réflexe avec état interne, but et utilité) et pourquoi ?
    a. Un agent contrôlant une valve de pression d'une centrale nucléaire.
    b. Un agent devant sortir d'un labyrinthe.
    c. Un agent conduisant un automobile.
    d. Un agent qui achète et vend des actions sur Internet.
    e. Un agent qui nettoie la vaisselle et qui la range dans les armoires.

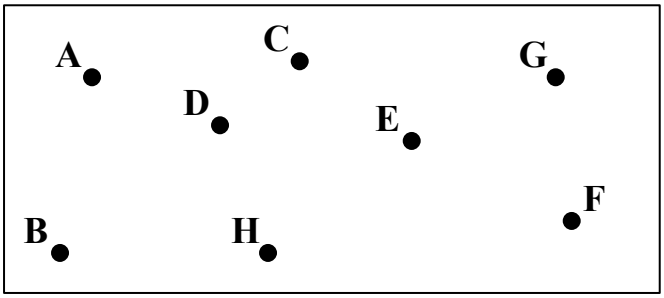
2. (22 pts) (35 min) Un fermier a une chèvre, un loup et une laitue sur la rive ouest d'une rivière. Il veut amener ses animaux et sa laitue de l'autre côté de la rivière, sur la rive est. Le fermier a un petit bateau à rame et il n'y a de la place que pour lui et une autre chose. De plus, le loup va manger la chèvre s'ils sont laissés seuls ensemble et la chèvre va manger la laitue si elle est laissée seule avec. Comment le fermier peut faire pour transporter tous les éléments sur la rive est ?
    a. Formuler ce problème comme un problème de recherche. C'est-à-dire, donner la représentation des états, l'état de départ, l'état but et les opérateurs pour passer d'un état à un autre.
    b. Résoudre ce problème de recherche (en utilisant la méthode de votre choix). Dessiner l'arbre de recherche et donner la solution finale.

3. (21 pts) (35 min) Derek est un ingénieur en robotique et il a conçu un robot qui doit se déplacer entre les édifices sur le campus pour délivrer le courrier. Il a implémenté l'algorithme A* pour la recherche de chemin (en utilisant la distance en ligne droite comme fonction heuristique), toutefois l'algorithme ne semble pas fonctionner correctement, car le robot choisit souvent un chemin qui n'est pas optimal. Voici le pseudo code qu'il utilise (où F est une file, D est le nœud de départ et h est la fonction heuristique) :

F = {*D*}
Tant que *F* n'est pas vide
    Prendre *F1*, le premier élément dans *F*
    Nœuds_enfants = développer(*F1*)
    Éliminer les nœuds dans *Nœuds_enfants* qui ont déjà été visités
    Pour tous les nœuds restant dans *Nœuds_enfants*, faire
        Si l'enfant est un état but
            Retourner Succès et Sortir
    Fin pour tous
    Ajouter les enfants à *F*
    Trier *F* selon la fonction suivante : *f = coûtChemin(D à nœud) + h(nœud)*
Fin tant que

a. Cet algorithme ne donne pas toujours la solution optimale, car il contient une erreur. Identifiez cette erreur et expliquez pourquoi la solution retournée n'est pas toujours optimale.
b. Corrigez alors l'erreur de Derek et écrivez le bon pseudo code pour A*.

4. (15 pts) (30 min) Anna est en charge de l'écriture d'un programme pour contrôler une machine d'assemblage. La machine est constituée d'un bras robotisé qui doit souder des puces électroniques sur une carte de circuit imprimé à certain points {A, B, C, D, E, F, G, H} comme montré sur la figure suivante :



Le bras commence dans le coin en haut à gauche et il doit visiter chaque point sur la carte de circuit imprimé et retourner à son point de départ. Aider Anna à minimiser la distance total que le bras doit emprunter en modélisant ce problème à l'aide d'algorithmes génétiques. Vous devez décrire tout ce qui est nécessaire aux algorithmes génétiques (les individus, les opérateurs génétiques, l'évaluation des individus, etc.) et expliquer le fonctionnement de l'algorithme pour trouver une bonne solution pour le problème.

5. (15 pts) (30 min) Considérons l'exemple de l'aspirateur illustré sur la figure suivante. Dans cet environnement, il y a un petit robot aspirateur qui doit nettoyer la maison. Le robot est équipé de détecteurs qui lui permettent de savoir s'il est par dessus de la saleté et d'un aspirateur qui lui permet d'aspirer la saleté. De plus, le robot a toujours une orientation bien définie (nord, sud, est ou ouest). En plus de pouvoir aspirer la saleté, le robot est capable d'avancer d'un pas ou de tourner à droite de 90°. L'agent se déplace dans une pièce divisée en case de grandeurs identiques. Le robot ne fait rien d'autre que nettoyer et donc il ne quitte jamais la pièce. Pour simplifier le problème, les dimensions de la pièce sont de 3x3 et le robot commence toujours dans la case (0,0) en regardant vers le nord.

En résumé, l'agent peut recevoir la perception *saleté* (signifiant qu'il y a de la saleté en dessous de lui), ou *rien* (indiquant aucune information spéciale). Il peut exécuter une des trois actions suivantes : *avancer*, *aspirer*, *tourner*. Le but de l'agent est de traverser toute la pièce en cherchant et en ramassant la saleté.

Le comportement de l'agent est géré par un ensemble de règles. Les trois prédicats simples suivants sont utilisés :

Pos(x,y)	l'agent est à la position *(x,y)*
Saleté(x,y)	il y a de la saleté à la position *(x,y)*
Orientation(d)	l'agent regarde dans la direction *d*

Le robot se promène toujours dans l'environnement en passant de (0,0) à (0,1) à (0,2) à (1,2), à (1,1) et ainsi de suite. Lorsque l'agent a atteint la case (2,2), il doit revenir à la case (0,0). Les règles pour permettre au robot de se déplacer sur la première ligne droite sont très simples :

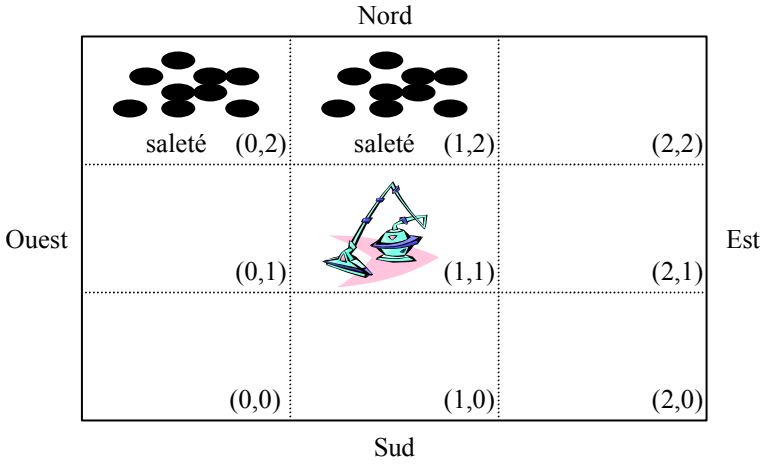
$Pos(0,0) \wedge Orientation(Nord) \wedge \neg Saleté(0,0) \rightarrow Faire(avancer)$

$Pos(0,1) \wedge Orientation\ (Nord) \wedge \neg Saleté\ (0,1) \rightarrow Faire(avancer)$

$Pos(0,2) \wedge Orientation\ (Nord) \wedge \neg Saleté\ (0,2) \rightarrow Faire(tourner)$

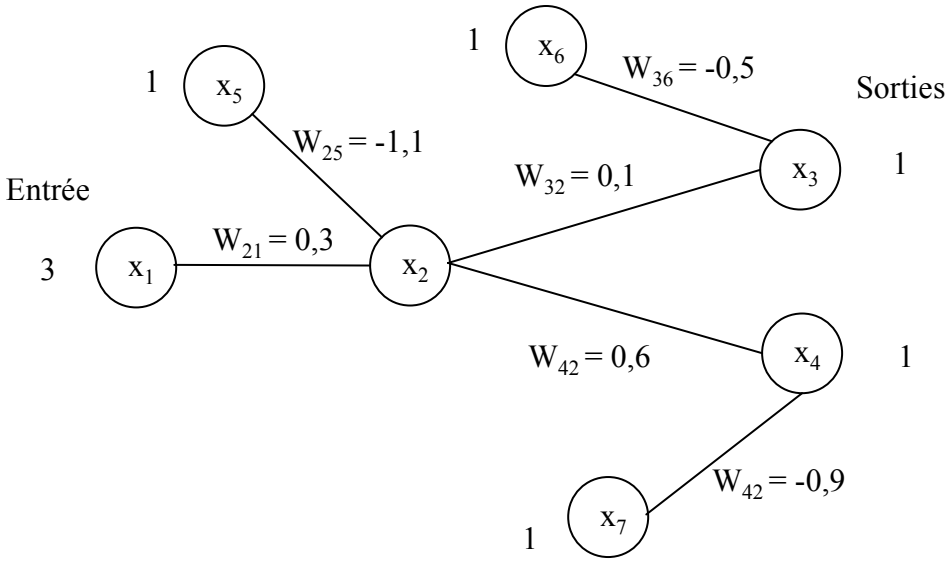
$Pos(0,2) \wedge Orientation\ (Est) \rightarrow Faire(avancer)$

Complétez la base de règles pour que l'agent puisse se rendre à la case (2,2) et ensuite revenir à la case (0,0) en nettoyant la salle.



6. (12 pts) (20 min) Effectuez un tour de l'algorithme de rétropropagation des erreurs et indiquez la valeur des nouveaux poids. Vous devez présenter tous les calculs. La valeur de la constante d'apprentissage est $\eta = 0{,}05$. La fonction d'activation est la fonction suivante :

$$f = \begin{cases} 1 \text{ si } \sum_{i=0}^{n} w_i \cdot x_i > 0 \\ -1 \text{ sinon} \end{cases}$$



Annexe : Algorithme de rétropropagation des erreurs

Pour chaque exemple d'entraînement

- Calculer les sorties du réseau
- Pour toutes les unités de sortie *k*, calculer l'erreur δ_k de la façon suivante:

$$\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$$

- Pour toutes les unités cachées *h*, calculer l'erreur δ_h de la façon suivante:

$$\delta_h \leftarrow o_h(1-o_h)\sum_{k\in sorties} w_{kh}\delta_k$$

- Mettre à jour tous les poids w_{ji} de la façon suivante:

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

ou

$$\Delta w_{ji} = \eta \delta_j x_{ji}$$

X_{ji} : l'entrée qui provient de l'unité *i* vers l'unité *j*.

W_{ji} : le poids du lien entre l'unité *i* et *j*.

T_k : la sortie attendue du réseau.

O_k : la sortie obtenue du réseau.