Représentation des connaissances

Evaluation Ecrite (session 1)

Durée : 2h
Matériel autorisé : une feuille A4 recto/verso de notes personnelles

1 Algorithme de recherche RBFS [8 pt]

D'autres algorithmes sous contrainte de mémoire existent comme l'algorithme de recherche recursive du meilleur d'abord (en anglais Recursive Best First Search ou RBFS) décrit par l'algorithme 1.

Algorithme 1 Algorithme de recherche récursive du meilleur d'abord. L'appel de la fonction est RBFS(probleme, CREER-NOEUD(probleme.ETAT-INITIAL), ∞)

```
Fonction RBFS(probleme, nœud, f_limite) retourne une solution ou (échec, une nouvelle limite de f)

si probleme.TEST-BUT(nœud.ETAT) alors
  retourner SOLUTION(nœud)
fin si
successeurs ← ∅
pour tout action ∈ probleme.ACTIONS(nœud.ETAT) faire
  ajouter NOEUD-FILS(probleme, nœud, action) à successeurs
  si successeurs est vide alors
    retourner (échec, ∞)
  fin si
fin pour
pour tout s ∈ successeurs faire
  s.f ← max (s.g+s.h, noeud.f)
fin pour
tant que Vrai faire
  meilleur ← la plus petite valeur f de nœud dans les successeurs
  si meilleur.f > f_limite alors
    retourner (échec, meilleur.f)
  fin si
  alternative ← la seconde plus petite valeur f parmi les successeurs
  (resultat, meilleur.f) ← RBFS(probleme, meilleur, min(f_limite, alternative)
  si resultat ≠ échec alors
    retourner resultat
  fin si
fin tant que
```

Appliquer l'algorithme RBFS au problème de la recherche d'un itinéraire d'Arad à Bucarest (voir figure 1).

Remarques :

- Pour faciliter la représentation de l'arbre de recherche vous pouvez n'utiliser que la première lettre des villes.
- Faites figurer la valeur f et sa limite actuelle auprès de chaque nœud.
- Vous trouverez le coût réel des distances sur la figure 1. et la valeur de l'heuristique pour aller à Bucarest dans la table 1.
- Vous pouvez vous contenter de représenter les étapes clefs qui montrent l'effet de la contrainte de mémoire.

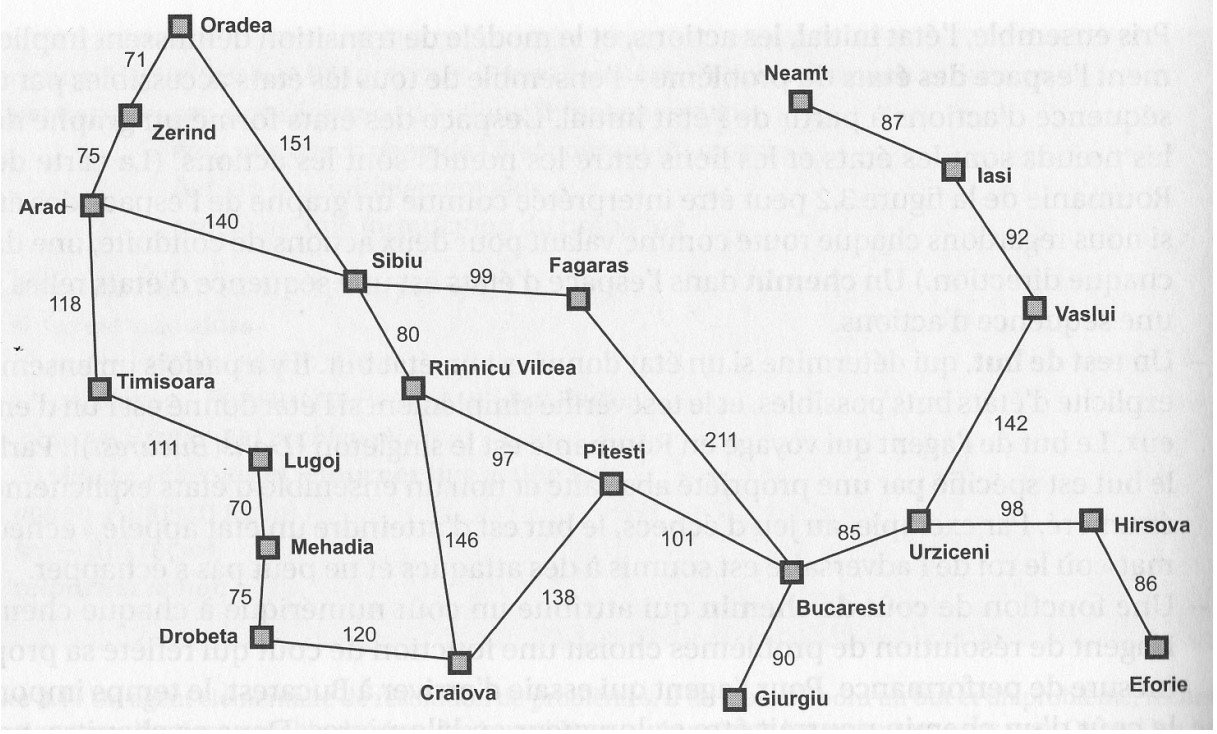

Figure 1: Carte routière simplifiée d'une partie de la Roumanie

Table 1: Estimation de la distance à vol d'oiseau jusqu'à Bucarest.

Ville	km	Ville	km	Ville	km	Ville	km
Arad	366	Fagaras	176	Mehadia	241	Sibiu	253
Bucarest	0	Giurgiu	77	Neamt	234	Timisoara	329
Craiova	160	Hirsova	151	Oradea	380	Urziceni	80
Dobreta	242	Iasi	226	Pitesti	100	Vaslui	199
Eforie	161	Lugoj	244	Rimnicu Vilcea	193	Zerind	374

2 Alpha-Beta [5pt]

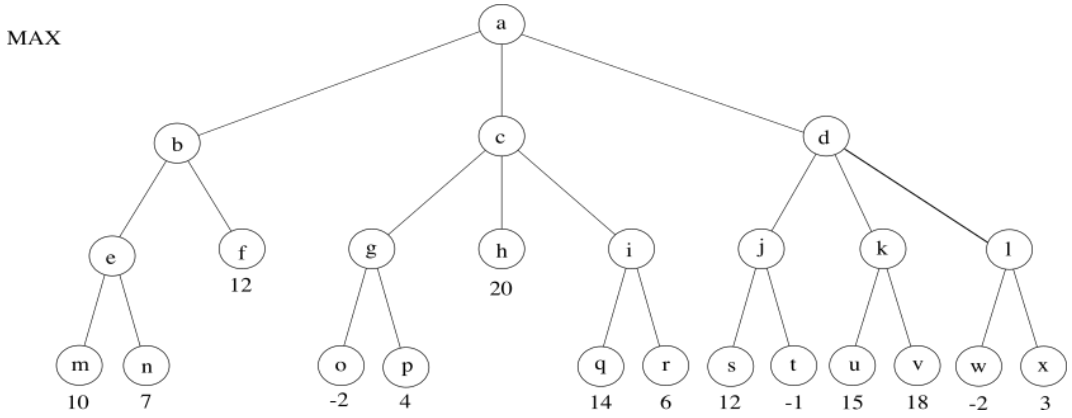

Figure 2: Arbre de jeu. La racine est un nœud maximisant. Chaque lettre identifie un nœud. Chaque chiffre correspond à l'évaluation d'une feuille.

1. Appliquer l'algorithme α-β à l'arbre de jeu donné par la figure 2.
   Indiquer tous les nœuds qui ne sont pas considérés par l'algorithme.

2. Dans quel état devrions nous arriver ?

3 Aide à la découverte de gisements d'or [5pt]

Le bureau de recherches géologique et minière en charge de la gestion durable des ressources naturelles souhaite mieux comprendre la formation de l'or et mieux prédire la présence ou l'absence de ce minerai.
Pour cela, il a constitué à partir d'un système d'information géographique une banque de données constituée de 1000 individus décrits par 24 variables, dont 21 sont quantitatives (20 variables géophysiques et le numéro du site) et 3 sont qualitatives (l'âge de la roche à 3 modalités, le type de la roche à 2 modalités et *site* à 2 modalités). On cherche en particulier à prédire efficacement la valeur de la variable qualitative *site* qui informe de la présence ("gisement") ou de l'absence ("stérile") du minerai pour chacun des 1000 sites connus.

		Classes prédites	
		gisement	stérile
Classes réelles	gisement	87	100
	sterile	0	813

Table 2: Matrice de confusion obtenue sur le corpus de test des gisements d'or.

Algorithme 2 Alpha-Beta

```
α-β(Node s, int alpha, int beta) //La valeur d'un nœud est encadrée par alpha et beta.
si isTerminal(s) alors
  return (GainEvaluation(s))
sinon
  // appliquer alpha-beta sur les fils de s et mémoriser leurs valeurs
  s.alpha = -∞;
  s.beta = +∞;
  si isComputerTurn(s) alors
    pour tout successeur si de s faire
      val = α-β(si, max(alpha, s.alpha),beta)
      s.alpha = max(s.alpha, val)
      si s.alpha ≥ beta alors
        break
      fin si
    fin pour
    return s.alpha
  sinon
    // tour de l'adversaire
    pour tout successeur si de s faire faire
      val = α-β(si, alpha, min(beta, s.beta))
      s.beta = min(s.beta, val)
      si s.beta ≤ alpha alors
        break
      fin si
    fin pour
    return s.beta
  fin si
fin si
```

1. De quelle tâche s'agit-il (régression, discrimination, classification, classement) ? Justifiez.

2. Quel réseau de neurones peut être utilisé pour modéliser ce problème ? Précisez :
   - le nom du réseau et le nom de l'algorithme associé ;
   - le nombre d'entrées et les transformations associées ;
   - le nombre de sorties.

3. Calculez le taux d'erreur au regard de la table 2.

4. A quelle(s) valeur(s) doit-on comparer le taux d'erreur pour savoir si le modèle est bon ?

5. Calculez le rappel pour la classe *gisement*.

6. Calculez la précision pour la classe *gisement*.

7. Ce modèle est-il utile ? Justifiez.

4 Auto-organisation des couleurs [2pt]

La chef de rayon d'une enseigne de bricolage doit réaménager son rayon peinture. Elle dispose d'une cinquantaine de teintes différentes à ranger dans un rayonnage avec 6 étages et 12 emplacements par étage.

1. Expliquez comment l'utilisation d'une carte auto-organisatrice de Kohonen peut lui permettre de ranger les teintes proches à des emplacements proches.