

**NOM DU DOSSIER**

**Sous titre**

Date

Service

**RAPPORT DE PROJET**

**Antoine ALAVERDOV, Clémence LEMEILLEUR**

**Promo 56, Année 2021/2022 – 4IR-SI-B1**

*« TP IA»*

Mars 2022

Encadrant : P.COUPECHOUX

**RAPPORT DE PROJET**

Antoine ALAVERDOV, Clémence LEMEILLEUR

Promo 56, Année 2021/2022 – 4IR-SI-B1

*“TP IA*”

Mars 2022

Encadrant: A.BIT-MONNOT

SOMMAIRE :

1. **Déroulé des TP et réponses aux questions1**
2. **Code source et tests unitaires1**
3. Code source commenté1
4. Tests unitaires effectués2
5. **Temps de réponse du programme pour différents problèmes de difficulté variée1**
6. **Limitations du programme, adaptabilité à d’autres problèmes1**
7. **Extensions réalisées ou entrevues1**

**Table des illustrations13**

**Table des annexes14**

**I-** **Déroulé des TP et réponses aux questions**

ALGORITHME A\* - APPLICATION AU TAQUIN

**1 Familiarisation avec le problème du Taquin 3**×**3**

**1.2 a) Quelle clause Prolog permettrait de représenter la situation finale du Taquin 4x4 ?**

etat\_final\_4x4([ [1, 2, 3, 4],

[5, 6, 7, 8],

[9, 10, 11, 12],

[13, 14, 15, vide] ]).

**b) A quelles questions permettent de répondre les requêtes suivantes :**

**?- initial\_state(Ini), nth1(L,Ini,Ligne), nth1(C,Ligne, d).**

Recuperer la ligne et la colonne respectivement dans L et C de d dans l'etat initial

**?- final\_state(Fin), nth1(3,Fin,Ligne), nth1(2,Ligne,P)**

Recuperer la piece qui se trouve a la 3eme ligne et la 2eme colonne a l'etat final

**c) Quelle requête Prolog permettrait de savoir si une pièce donnée P (ex : a) est bien placée dans *U0* (par rapport à F) ?**

initial\_state(Ini), nth1(L,Ini,Ligne), nth1(C,Ligne,d), final\_state(Fin), nth1(L,Fin,LigneF), nth1(C,LigneF,d).

**On s’intéresse maintenant au prédicat rule/4. Comment l’utiliser pour répondre aux questions suivantes :  
d) quelle requête permet de trouver une situation suivante de l’état initial du Taquin 3×3 (3 sont possibles) ?**

initial\_state(Ini), rule(\_,\_,Ini,Next).

**e) quelle requête permet d'avoir ces 3 réponses regroupées dans une liste ? (*cf*. findall/3 en Annexe).**

findall(Next, (initial\_state(Ini), rule(\_,\_,Ini,Next)), R).

**f) quelle requête permet d'avoir la liste de tous les couples [A, S] tels que S est la situation qui résulte de l'action A en U0 ?**

findall([A, Next], (initial\_state(Ini), rule(A,\_,Ini,Next)), R).

**2. Développement des 2 heuristiques**

2.1 L’heuristique du *nombre de pièces mal placées*.

On a choisi la methode a : rechercher la liste de toutes les *pièces mal placées* (cf. prédicat **findall**) et calculer sa taille (cf. prédicat **length**). Une pièce (attention ‘**vide**’ n’en est pas une) est *mal placée* en U si le terme de mêmes cordonnées dans l’état final F est un terme différent. Le prédicat liant un terme à ses coordonnées dans une matrice doit donc être défini.

malplace(P, U, Final):-

nth1(L,U,Ligne), nth1(C,Ligne,P), nth1(L,Final,LigneF),

nth1(C,LigneF,M), P\=M, P\=vide.

heuristique1(U, H) :-

final\_state(Final), findall(P, malplace(P, U,Final), Res),

length(Res, H).

2.2 L’heuristique basée sur la *distance de Manhattan*.

A FAIRE

**3. Implémentation de A\***

**3.1 Implémentation de P et Q par des arbres AVL**

A FAIRE

**3.2 Algorithme A\* adapté aux structures AVL choisies**

A FAIRE

**3.3 Analyse expérimentale**

A FAIRE

ALGO MINMAX - APPLICATION AU TICTACTOE

**1 Familiarisation avec le problème du TicTacToe 3**×**3**

**1.2 Compiler ce programme et répondre aux questions suivantes :  
Quelle interprétation donnez-vous aux requêtes suivantes :**

**?- situation\_initiale(S), joueur\_initial(J).**

Les numeros correspondent aux cases non instanciées et J est le joueur à qui c'est le tour

**?- situation\_initiale(S), nth1(3,S,Lig), nth1(2,Lig,o)**

Met un 'o' a la deuxieme case de la 3eme ligne.

**2.2 Comment accéder à une ligne, une colonne, une diagonale d’une matrice matrice carrée NxN ?**

A FAIRE

**Définir le prédicat possible(Ali, Joueur)**

A FAIRE

**Définir enfin les prédicats alignement\_gagnant(A, J) et alignement\_perdant(A, J) qui réussissent si A est un alignement totalement instancié (utiliser le prédicat ground pour le savoir) ne contenant que des valeurs J (respectivement que des valeurs de l’adversaire de J).**

A FAIRE

**Proposer des requêtes de tests unitaires pour chaque prédicat.**

A FAIRE

**2. Développement de l’heuristique *h*(*Joueur, Situation*)**

**Développer le prédicat heuristique(Joueur,Sit,H) qui retourne la valeur de l’heuristique pour le joueur J dans une situation donnée.**

A FAIRE

**Proposer une requête permettant de tester votre heuristique dans la situation initiale (elle doit retourner 0 quel que soit le joueur).**

A FAIRE

**Proposer d’autres tests unitaires pour vérifier qu’elle retourne bien les valeurs attendues dans le cas d’une situation gagnante pour J, perdante pour J ou nulle (toutes les cases ont été jouées sans qu’aucune joueur n’ait gagné).**

A FAIRE

**3. Développement de l’algorithme Negamax**

**3.2 Quel prédicat permet de connaître sous forme de liste l’ensemble des couples [Coord, Situation\_Resultante] tels que chaque élément (couple) associe le coup d’un joueur et la situation qui en résulte à partir d’une situation donnée.**

A FAIRE

**Tester ce prédicat en déterminant la liste des couples [Coup, Situation Resultante] pour le joueur X dans la situation initiale.**

A FAIRE

**3.3 Compléter les définitions demandées et tester-les en proposant des tests unitaires.**

A FAIRE

**4. Expérimentation et extensions**

**4.1 Quel est le meilleur coup à jouer et le gain espéré pour une profondeur d’analyse de 1, 2, 3, 4 , 5 , 6 , 7, 8, 9 Expliquer les résultats obtenus pour 9 (toute la grille remplie).**

A FAIRE

**4.2 Comment ne pas développer inutilement des situations symétriques de situations déjà développées ? 4.3 Que faut-il reprendre pour passer au jeu du puissance 4 ?**

A FAIRE

**4.4 Comment améliorer l’algorithme en élaguant certains coups inutiles (recherche Alpha-Beta) ?**

A FAIRE

**II-** **Code source et tests unitaires**

1. *Code source commenté*

A mettre

1. *Tests unitaires effectués*

A détailler un peu

**III-** **Temps de réponse du programme pour différents problèmes de difficulté variée**

A mettre

**IV-** **Limitations du programme, adaptabilité à d’autres problèmes**

A mettre

**V-** **Extensions réalisées ou entrevues**

A mettre

***Conclusion :***

Tout au long de ces TP, nous avons réussi à développer répondre aux problèmes posés en codant au fur et à mesure les prédicats en Prolog. Nous avons réussi à gérer l’approche des différents problèmes de jeu de taquin, morpion par exemple, puis le codage et enfin les tests unitaires. Nous avons pu expérimenter et mieux comprendre les principes de base de l’intelligence artificielle. Cela nous permet donc de nous mettre une fois de plus dans le rôle de l’ingénieur qui est d’utiliser ses connaissances et de les appliquer à des cas réels.

Table des illustrations

**Figure 1.** – Screenshot personnel [25/02/2022]. *Légendes des diagrammes*.

Table des annexes

1. Annexe 1 : Lien GitHubA

**Annexe 1 :** Lien GitHub : https://github.com/Clemence-Lemeilleur/TP\_IA

