CARBONNIER Nicolas
WYSOCKI Tom
WATELET Manon

L2 – MIASHS
Sciences Coginitives
2018 - 2019

Institut des sciences du Digital
Management & Cognition

Projet de groupe – Rapport Intelligence Artificielle

FINSTERE FLURE

Objectif : Donner une stratégie de mise en œuvre par le programme pour gagner (ex. : **parcours de graphe** pour une **recherche du plus court chemin**, **gain attendu dans chaque état**, algorithmes **minimax**, **élagage** alpha-beta, **heuristiques**, etc.) avec représentation des données. Un joueur ordinateur peut également chercher à mettre en défaut un joueur adverse en attirant le monstre vers lui, ou avoir une stratégie plus ou moins risquée.

Tous les problèmes étudiés jusqu'à présent se présentaient sous cette forme : on avait un problème et une solution finale à laquelle aboutir (jeu du taquin, cannibales et missionnaires etc.). Pour cela plusieurs chemins sont parfois possibles et on utilisait 3 algorithmes piliers dans la résolution de ce problème :

- *Un* ***SSALGO But?*** *qui vérifie si l'état final e_f (=solution attendue) est atteint ou non après chaque opération effectuée*
- *un* ***SSALGO Successeur*** *qui génère tous les états e_{n+1} à partir d'un état e_n*
- *un* ***SSALGO Recherche*** *qui donne la liste des chemins qui mènent jusqu'à cet état final (selon le nombre de possibilités) on peut retourner : une liste vide, d'un seul élément ou plusieurs) autrement une la liste de toute les actions vers le résultats final attendu.*

Dans le cas du jeu ***Finstere Flure****, on a une omniscience sur la map, où tout est observable (pions alliés, adverses, monstre et la nature de la case – mur, flaque de sang ou occupée) mais on a un aspect indéterministe ou imprévisible pour les déplacements adverses et ceux du monstre.*

C'est pour cela que l'usage du ***SSALGO But?*** *est plus complexe à mettre en place et est différent de ceux qu'on a connu jusqu'à présent, dans la mesure où le* ***SSALGO Successeur*** *retourne tous les coups n+1 accessibles (au coup n) et que l'environnement évolue après chaque coup exécuté.*

Le ***SSALGO Recherche*** *retourne, quad à lui, la liste* ***lvictoire*** *de ceux qui favorisent une stratégie gagnante (en se rapprochant le plus de la sortie, en sortant, ou en sauvant un pion proche du monstre) à chaque étape.*

Pour cela nous avons dans un premier temps fait la liste des **types** et des **variables** (à l'aide de ce qui a été fait dans le code Java) pour connaître toutes les caractéristiques à prendre en compte dans la stratégie de Recherche. On a ensuite fait une sélection des Opérateurs utiles pour la recherche, avant de la mettre en place et de détailler comment elle va se faire.

Une **heuristique** n'est pas possible dans notre cas car on procède coup par coup et on effectue une recherche du meilleur coup à chaque fois. Il n'y a donc pas une recherche qui donne une séquence d'action à effectuer pour mener à la victoire à coup sûr, mais seulement des coups plus performants que d'autres.

Par soucis de temps et de complexité, nous n'avons pas abordés certains aspects bien que nous ayons conscience de l'enjeu qu'ils ont dans la recherche stratégique :

- On pourrait prendre en compte les cartes déjà utilisées pour anticiper les mouvements du monstre et savoir s'il sera au prochain tour en capacité d'atteindre une certaine case ou si les cartes qui lui reste l'empêchent de nous atteindre.
- Prendre en compte les pions adverses qui seront jouables ou non au prochains coups pour favoriser une stratégie ou non (comme le mettre en danger par rapport au monstre sachant qu'il ne pourra pas déplacer son pion avant la fin du tour)
- Prendre en compte davantage dans les SSALGO l'orientation du monstre pour savoir s'il représente un danger ou non.

Types

Tetat = ARTICLE
sorti : BOOLEEN // vrai si pion sauvé
mort : BOOLEEN // vrai si pion mort
dm : NUMERIQUE // distance par rapport au monstre
ds : NUMERIQUE // distance par rapport à la sortie
FART

Tmap = ARTICLE
labyrinthe : TABLEAU DE 16 TABLEAU DE 11 // tableau à double entrée
FART

Tpion = ARTICLE
e : Tetat
coord : TABLEAU DE 2 NUMERIQUE // coordonnées x et y dans le labyrinthe
FART

Tmonstre = ARTICLE
coord : TABLEAU DE 2 NUMERIQUE // coordonnées x et y
orient : NUMERIQUE
FART

Tcoup = ARTICLE
e : Tetat
p : Tpion
FART

Variables

p1,p2,p3,p4 : Tpion
m : Tmonstre

coord : TABLEAU DE 16 TABLEAU DE 11 // tableau à double entrée
orient : NUMERIQUE // -1 si vers bas, 1 si vers haut, -2 si vers la gauche, 2 si vers la droite
lsortie : Tliste de Tpion // pions sorti du tableau (sauvés)
lmort : Tliste de Tpion // pions morts
lsuccesseur : Tliste de Tcoup

lvictoire : Tliste de Tcoup // liste de coups qui mènent vers la victoire
ldéfaite : Tliste de Tcoup // liste de coups qui mènent vers la défaite
lneutre : Tliste de Tcoup // liste de coups qui ne font ni gagner ni perdre

Mise en place d'une stratégie

- L'objectif est de construire un algorithme de Recherche qui, grâce aux autres SSALGO (cf liste ci-dessous) permettra de jouer le meilleur coup possible parmi la liste de tous ceux accessibles au moment de jouer pour le bot.

- D'une part, on utilise la liste suivante des Opérations applicables pour un pion :
  - Tetat SSALGO **Déplacer** *(p : Tpion)*
    - *Cet algorithme retourne un nouvel état où les coordonnées du pion passé en paramètre sont modifié. De plus, il retourne un booléen permettant de savoir si cette action est valide ou non*
  - Tetat SSALGO **PousserMur**
  - Tetat SSALGO **Glissade**
  - BOOLEEN SSALGO **CaseOccupée** *(par monstre/pion/mur)*

- Autres SSALGO utile pour la mise en place de la recherche du meilleur coup à jouer :
  - Tliste SSALGO **PionDispo** *(liste de pion disponible au moment de jouer un coup)*
  - NUMERIQUE SSALGO **Distance**

- SSALGO pour la stratégie de Recherche :
  - BOOLEEN SSALGO **But?** *(qui retourne un booléen VRAI si une solution intéressante est trouvée)*
  - Tliste SSALGO **Recherche**
  - Tliste SSALGO **Successeur**

Définition d'un état :

Un état e_n du bot à un instant n est définit par :

- *Nombre pions sortis*
- *Nombre pions morts*
- *Position par rapport au monstre*
- *Position par rapport à la sortie*

La performance d'un coup sera caractérisé par ces 3 éléments ; le coup qui sera donc joué pour atteindre le coup s_{n+1} sera celui qui aura le plus de pions sorti, le moins de pions mort et la distance par rapport au monstre la plus grande. On hiérarchise ces caractéristiques comme précédemment (i.e. si un coup fait sortir un pion au coup s_{n+1} mais qu'un autre fait

en sorte de s'éloigner du monstre au coup s_{n+1}' dans ce cas on privilégie celui qui fait sortir le pion).

Description du SSALGO **But?** :

Le SSALGO But? retourne un booléen VRAI *si l'une de ces actions est validés :*

Un pion est sauvé OU SI *la distance par rapport à la sortie diminue du coup s_n à s_{n+1}* OU SI *la distance par rapport au monstre est la plus grande (par rapport aux autre pions disponibles).*

```
BOOLEEN    SSALGO     But?  (p : Tpion PAR REFERENCE)

Variable

     lmort : Tliste de Tpion
     ds1 : NUMERIQUE //distance par rapport à la sortie au coup n
     ds2 : NUMERIQUE //distance par rapport à la sortie au coup n+1
     dm1 : NUMERIQUE //distance par rapport au monstre au coup n
     dm2 : NUMERIQUE //distance par rapport au monstre au coup n+1

Début

 RETOURNE( Appartient(lmort, p) OU (p.ds1-p.ds2)>0 OU (p.dm1-p.dm2)<0 )

Fin
```

Description du SSALGO **Successeur** :

*Ce SSALGO retourne dans **lsuccesseur** la liste de tous les coups s_{n+1} possibles d'obtenir à partir d'un état s_n en appliquant toutes les actions possibles sur tous les pions possibles (=disponibles).*

*Si un coup fait placer un pion à 5 points de mouvement du monstre, on l'ajoute à la liste **ldéfaite**.*

*On teste chaque Operateur sur chaque pion jouable. S'il est jouable, on vérifie s'il a 5 points de mouvement du monstre, si oui on l'ajoute à la liste **ldéfaite**, sinon on l'ajoute à **lsuccesseur.***

! Remarque : il est parfois possible d'être à moins de 5 points de mouvement du monstre et de pouvoir être protégé par un mur ou un autre pion plus proche et donc ne pas pour autant être en danger, mais ces cas-là ne sont (malheureusement) pas considérés ici.

```
l_successeur     SSALGO     Successeur (p : T_pion , cp : T_coup)

Variable

      l_sortie : T_liste de T_pion
      l_mort : T_liste de T_pion
      l_successeur : T_liste de T_coup
      l_défaite : T_liste de T_coup
      ds1 : NUMERIQUE //distance par rapport à la sortie au coup n
      ds2 : NUMERIQUE //distance par rapport à la sortie au coup n+1
      dm1 : NUMERIQUE //distance par rapport au monstre au coup n
      dm2 : NUMERIQUE //distance par rapport au monstre au coup n+1

Début
/
Fin
```

Description du SSALGO **Recherche** :

*Ce SSALGO retourne la liste de tous les coups pour lequel le booléen **but** retourne VRAI. L'action qui sera alors choisie sera prise dans la liste **lvictoire** de ceux rencontrant l'une des 3 conditions. Et c'est dans cette liste qu'on choisit l'action à jouer en recherchant le coup qui contient un état permettant d'arriver à sortir un pion ; s'il n'y en a pas on choisit celui qui retourne VRAI à **but** pour la 2ème condition et sinon la 3ème. Si la liste **lvictoire** est vide, on considère qu'aucune action n'est performante est qu'elle se valent toutes (on en exécute une de la liste **lneutre** au hasard).*

A ce moment-là on a :

***Lsortie** qui contient les pions sauvé (peut être vide)*
***lmort** qui contient les pions mort (peut être vide)*
***lsuccesseur** qui contient tous les coups possible (sauf si dans la liste ldéfaite*
***lvictoire** qui est vide pour l'instant*
***ldéfaite** qui peut contenir certain coups défavorisant*

```
lvictoire      SSALGO      Recherche (lsuccesseur PAR REFERENCE)

Variable

      lvictoire : Tliste de Tcoup
      lneutre : Tliste de Tcoup
      coup : Tcoup

Début

  lvictoire <- CREER()
  lneutre <- CREER()
  TANT QUE (NON VIDE(lsuccesseur)) FAIRE
      coup <- TETE(lsuccesseur)
      lsuccesseur <- RESTE(lsuccesseur)
      SI (But?(coup)) ALORS
            lvictoire <- ADJ (lvictoire , coup)
            SINON
            lneutre <- ADJ (lneutre , coup)
      FSI
  FTQUE
  SI (NON VIDE (lvictoire) ALORS
      RETOURNE (lvictoire)
      SINON
      RETOURNE (lneutre)
  FSI

Fin
```