

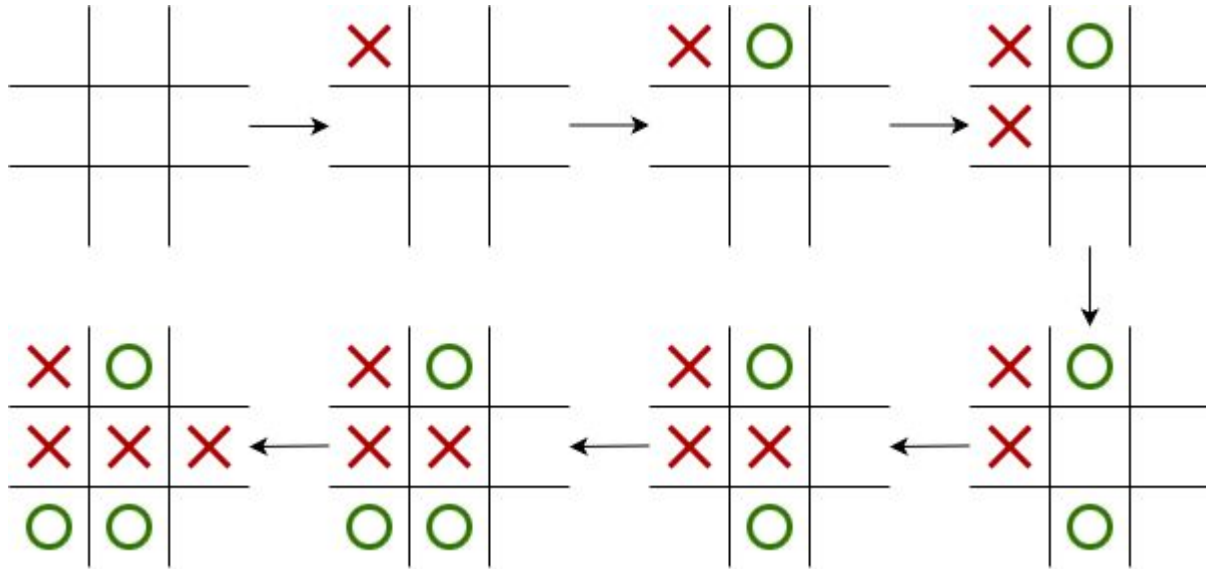
Projet 3

Intelligence artificielle pour le morpion

INFO0952

26/11/2024

Le morpion (aka tic-tac-toe, oxo)



Intelligence artificielle pour le morpion

Deux agents à implémenter:


- **Algorithme minimax**: détermine de manière exacte le coup optimal en faisant l'hypothèse que l'adversaire joue également de manière optimale
- **Apprentissage par renforcement**: adapte les coups au comportement de l'adversaire

Algorithme minimax

On attribue un score/récompense à chaque partie: **+1** si la partie est gagnée, **-1** si la partie est perdue, **0** sinon.

A partir d'une grille b , on détermine le **score maximum**, noté $S(b)$, que le joueur *dont c'est le tour* pourra obtenir en supposant que son adversaire tente également de maximiser son score.

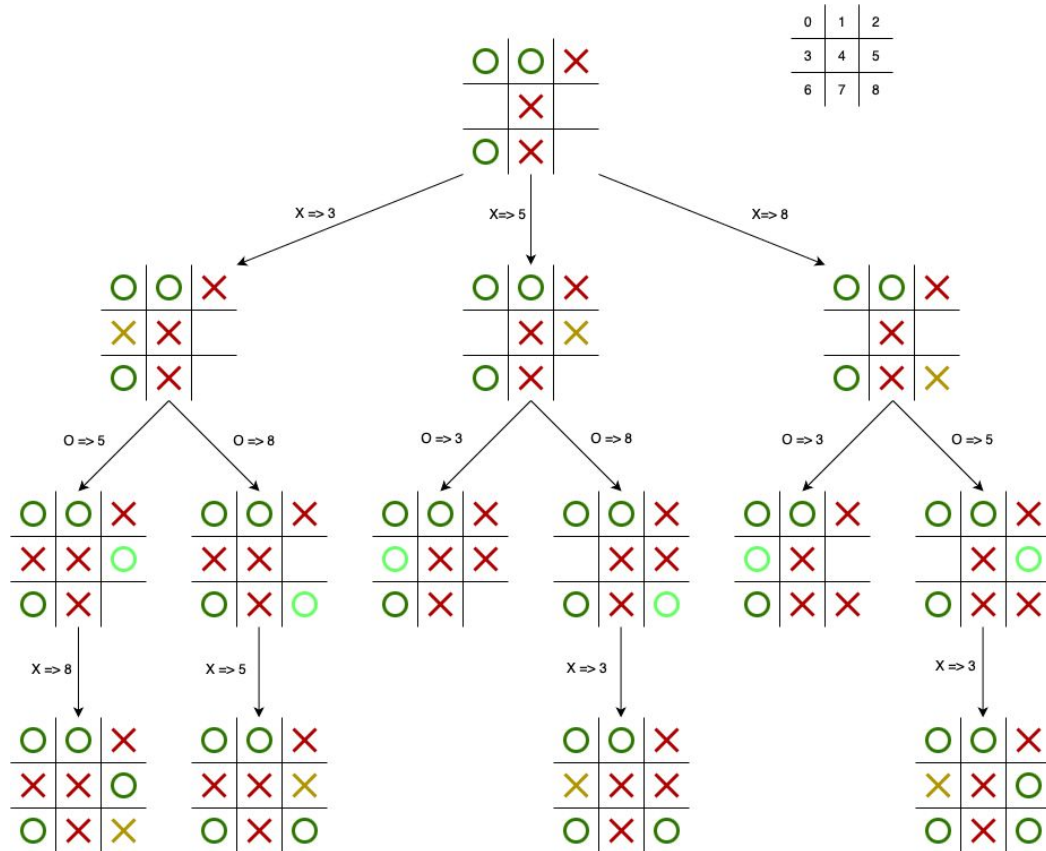
Formulation récursive:

$$S(b) = \max_{m \in \mathcal{M}(b)} -S(\text{next}(b, m))$$


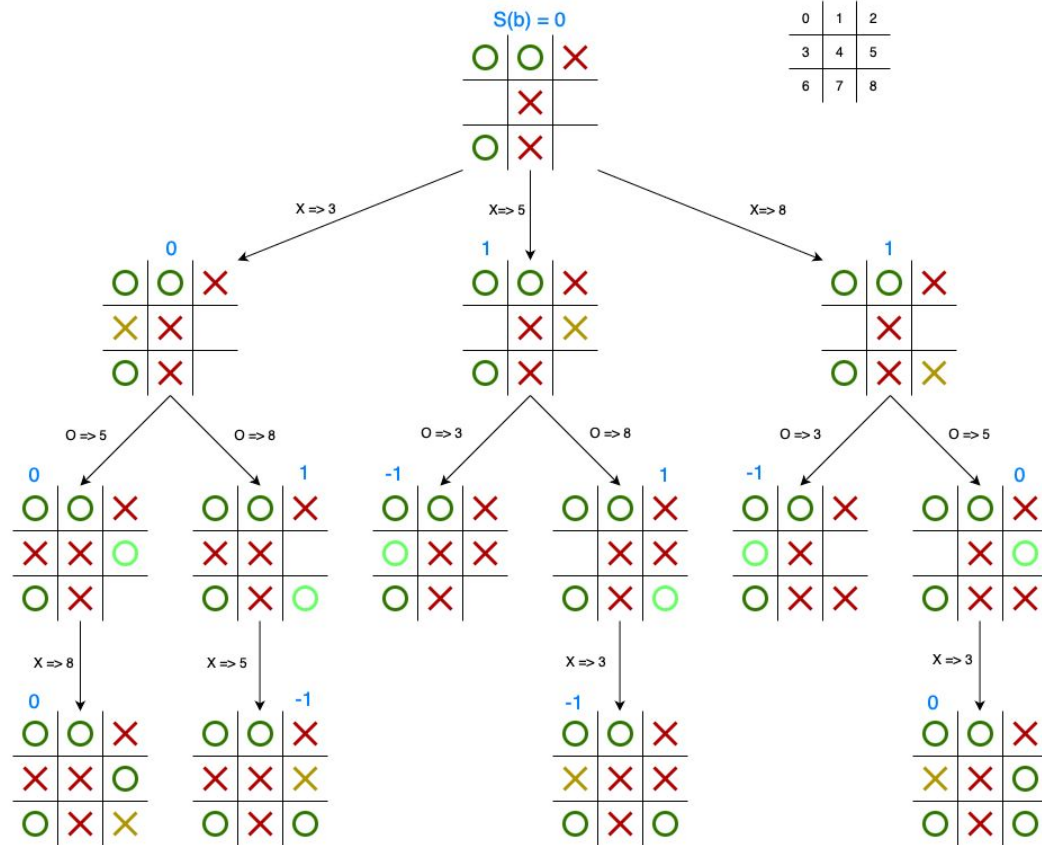
Tous les coups possibles à partir de b

La grille après le coup m

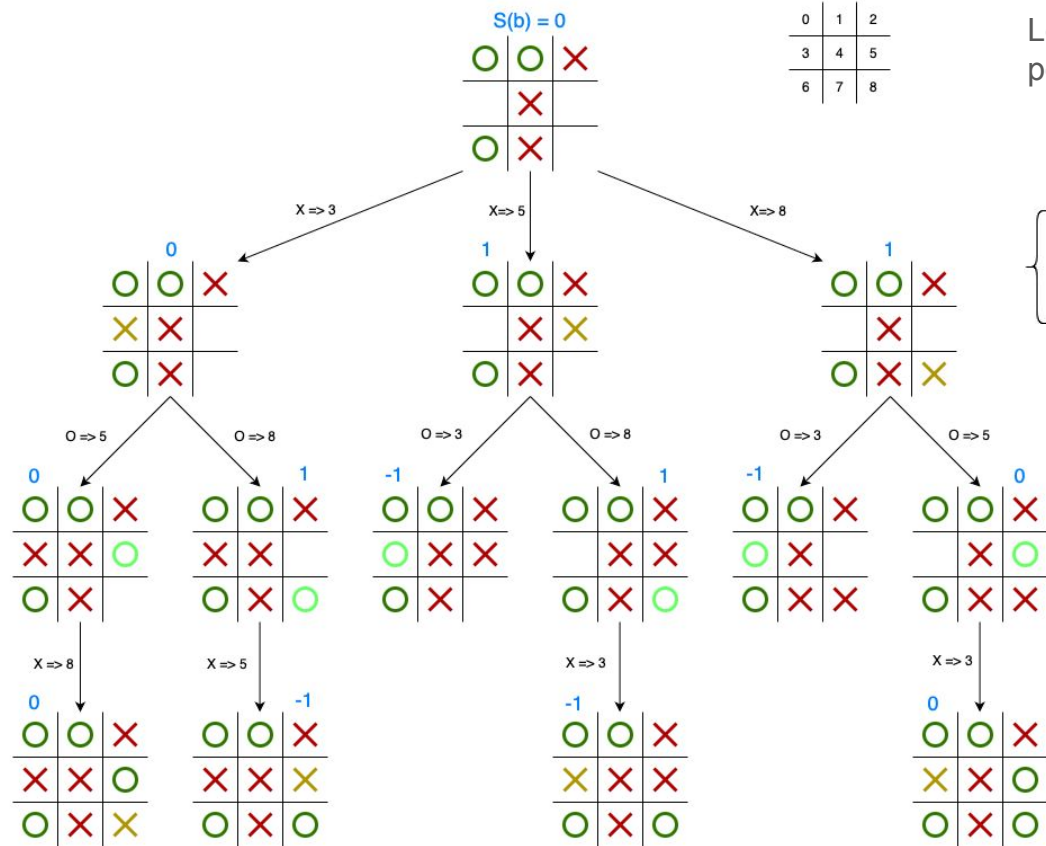
Algorithme minimax: illustration



Algorithme minimax: illustration

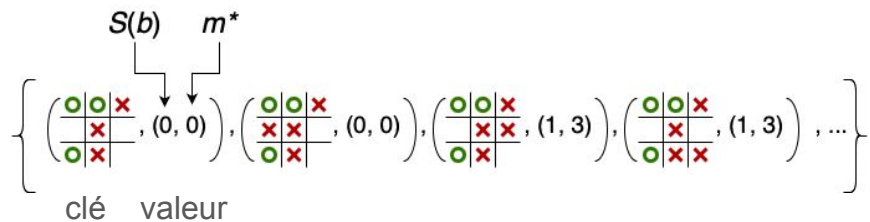


Algorithme minimax: illustration



0	1	2
3	4	5
6	7	8

Les scores sont stockés dans une table de hachage, pour éviter de les recalculer plusieurs fois



Apprentissage par renforcement

Idée générale:

- On estime le score moyen $\hat{S}(b, m)$ qu'on obtient quand on fait le coup m pour une grille b en observant des parties contre son adversaire
- On choisit le coup en fonction de ce score moyen (plus il est élevé, meilleur est le coup)

Deux modes:

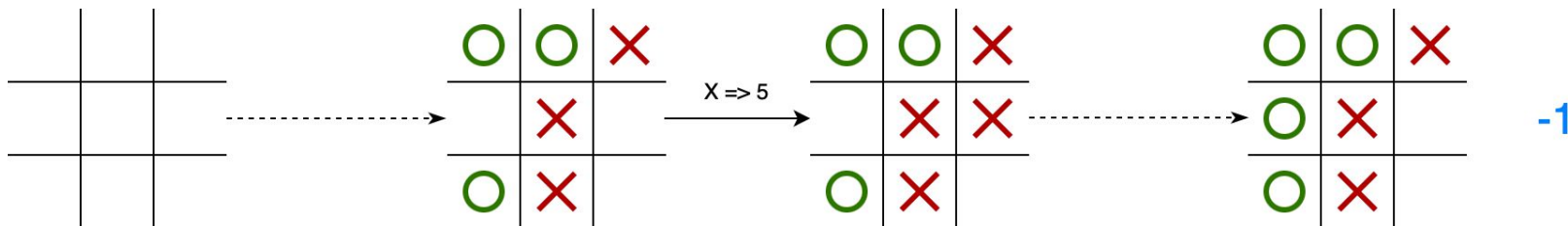
- Mode apprentissage: (1) on joue en choisissant un coup au hasard avec une probabilité ϵ et le coup qui maximise $\hat{S}(b, m)$ avec une probabilité $(1-\epsilon)$, (2) à chaque fin de partie, on met à jour $\hat{S}(b, m)$.
- Mode exploitation: on choisit le coup qui maximise $\hat{S}(b, m)$.

Apprentissage par renforcement: illustration

$$\hat{S}(b, m) = \frac{S_{tot}(b, m)}{N}$$

$$\left\{ \dots, \left(\begin{array}{c|c|c} \textcircled{0} & \textcircled{0} & \times \\ \hline & \times & \\ \hline \textcircled{0} & \times & \end{array} \right), \left\{ (3, 0, 1) \rightarrow (5, 0, 1) \rightarrow (8, 0, 1) \right\} \right\}, \dots \left\{ \right.$$

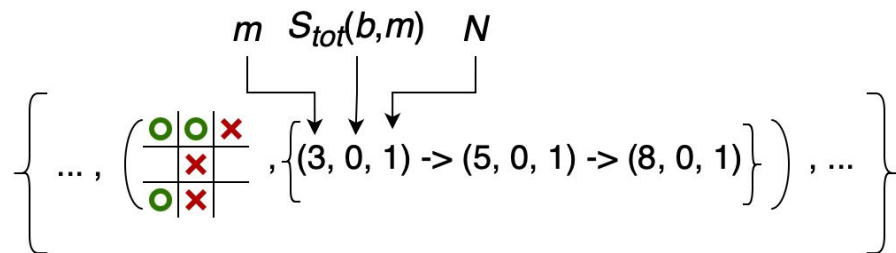
m $S_{tot}(b, m)$ N



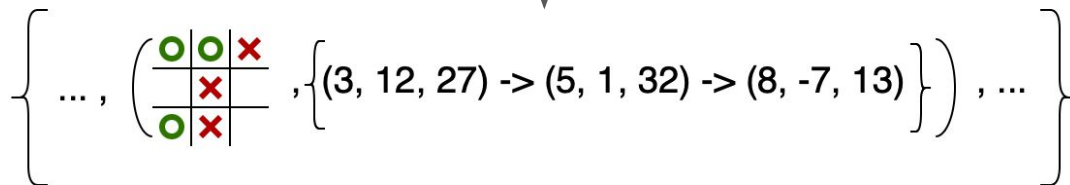
$$\left\{ \dots, \left(\begin{array}{c|c|c} \textcircled{0} & \textcircled{0} & \times \\ \hline & \times & \\ \hline \textcircled{0} & \times & \end{array} \right), \left\{ (3, 0, 1) \rightarrow (5, -1, 2) \rightarrow (8, 0, 1) \right\} \right\}, \dots \left\{ \right.$$

Apprentissage par renforcement: illustration

$$\hat{S}(b, m) = \frac{S_{tot}(b, m)}{N}$$



1000 parties, contre un agent aléatoire, avec $\varepsilon = 0.5$



Fichiers

Fournis:

- `board.c/board.h`: toutes les fonctions de manipulation d'une grille
- `agent.c/agent.h`: gestion des agents, fonction de jeu, agent humain et agent aléatoire
- `dict.c/dict.h`: table de hachage (clé = `char *`, valeur = `void *`)
- `LinkedList.c/LinkedList.c`: implémentation d'une liste liée
- `main.c`: permet de jouer

A implémenter:

- `aiagent.c/aiagent.h`: l'agent utilisant l'algorithme minimax
- `rlagent.c/rlagent.h`: l'agent utilisant l'apprentissage par renforcement

Création d'un agent

```
Agent *agentCreate(char *name, Move (*play) (Agent *, Board),  
                  void (*end) (Agent *, Board, Player),  
                  void (*freeData) (void *));
```

`name` est le nom de l'agent, `play` est appelée pour choisir un coup, `end` est appelée sur la dernière grille, `freeData` est appelée pour libérer les données.

```
void agentSetData(Agent *, void *);
```

```
void *agentGetData(Agent *);
```

permettent de stocker et retrouver des données associées à l'agent.

Création d'un agent: agent aléatoire (voir agent.c)

```
static Move randomAgentPlay(Agent *agent, Board b) {
    (void)agent;

    int moves[9] = {0};
    int numLegalMoves = 0;
    for (int m = 0; m < 9; m++) {
        if (boardValidMove(b, m)) {
            moves[numLegalMoves++] = m;
        }
    }
    return moves[rand() % numLegalMoves];
}

static void noEnd(Agent *agent, Board b, Player winner) {
    (void)agent; (void)b; (void)winner;
}

static void noFree(void *data) {
    (void)data;
}

Agent *createRandomAgent(void) {
    return agentCreate("Random agent", randomAgentPlay, noEnd, noFree);
}
```

Testez votre code

`./tictactoe -s human human`

`./tictactoe -s human ai`

`./tictactoe -s human rl`

`./tictactoe -s human rl -p 10000 ai`

`./tictactoe -t 1000 random ai`

`./tictactoe -t 1000 random rl -p 100000`

`./tictactoe -t 1000 random rl -p 100000 ai`