Introduction à la théorie des jeux

Ce chapitre permet de réviser les notions de programmation et de représentation des données par un graphe, en les appliquant à des enjeux contemporains. La connaissance dans le détail des algorithmes de cette section n'est pas un attendu du programme. Les étudiants se familiarisent avec les idées sous-jacentes, pour les réinvestir dans des situations où les modélisations et la programmation sont guidées.

1. Position du contexte au programme pour les jeux à deux joueurs. Stratégie, stratégie gagnante. Position gagnante.
2. Détermination des positions gagnantes par le calcul des attracteurs. Construction de stratégies gagnantes.
3. Algorithme min-max avec une heuristique.

Si le mot *jeux* a un côté amusant, le domaine de la théorie des jeux, depuis les travaux de von Neumann (1920) et Nash (1950), est un thème de travail sérieux avec des applications en économie, politique, physique, biologie,...
Ce chapitre est une introduction à la théorie des jeux à deux joueurs, J_1 et J_2. Les jeux que nous allons étudier sont :

— à information complète : à tout instant d'une partie, chacun des joueurs a une vision complète de l'état du jeu (au contraire des jeux de cartes, par exemple),

— sans hasard (au contraire des jeux de dés),

— alternativement joués par l'un puis l'autre des joueurs (ce qui exclut les jeux comme pierre-feuille-ciseaux).

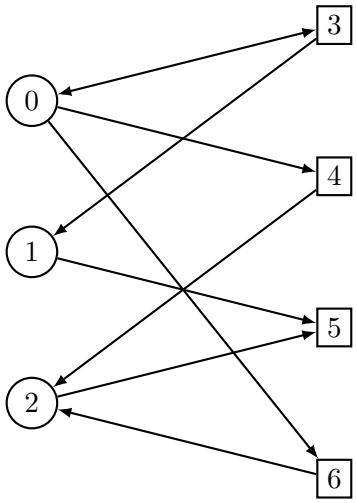
Nous commençons en partie 1 par étudier des jeux suffisamment simples pour être résolus complètement (l'espace des états n'étant pas trop grand). Puis en partie 2, on aborde la notion de stratégie avec heuristique, avec notamment l'algorithme Min-Max ou MiniMax, qui peut s'appliquer aux jeux qu'il est impossible de résoudre complètement à l'heure actuelle, comme le jeu d'échecs.

1 Jeux d'accessibilité à deux joueurs sur un graphe

1.1 vocabulaire sur les graphe

Dans tout le chapitre, on considère des graphes orientés.

Graphe biparti : un graphe est biparti si l'ensemble de ses sommets peut être divisé en deux sous-ensembles disjoints S_1 et S_2 tels que chaque arête de G a une extrêmité dans S_1 et une extrêmité dans S_2.
S_1 sera l'ensemble des sommets du joueur J_1 et S_2 l'ensemble des sommets du joueur J_2. Jouer un coup consiste à suivre un arc et les joueurs jouent alternativement. Une partie est un chemin (éventuellement infini) dans le graphe.



Contrôle : les sommets de S_1 sont contrôlés par le joueur J_1 (J_1 joue à partir de ces sommets) et les sommets de S_2 sont contrôlés par le joueur J_2. On les représente différemment (cercles/carrés).
Dans l'exemple, $S_1 =$ *et* $S_2 =$

Accessibilité : on parle de jeu d'accessibilité car tout sommet du graphe biparti correspond à une configuration atteignable dans le jeu par l'un des deux joueurs.
5 est accessible d'au moins deux façons :

État de départ / sommet s_0 : on considère généralement que c'est le joueur J_1 qui commence donc le sommet s_0 de départ est dans S_1.
Le sommet de départ peut être ici $s_0 = 0$.

Condition de victoire : le joueur J_1 gagne lorsqu'il atteint un sous-ensemble W de sommets de G prédéterminé. Il y a aussi un sous-ensemble de sommets constitués des états gagnants pour l'autre joueur, et un sous-ensemble N de sommets constitué des états de partie nulle.

Jeu à somme nulle : toute partie gagnée par l'un des deux joueurs est perdue par l'autre joueur.
Si $W = \{5\}$*, on conseille à* J_1 *de jouer :*

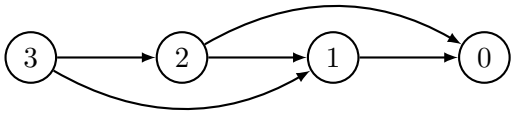
En résumé,

— se donner un *jeu d'accessibilité à deux joueurs* revient à se donner (G, S_1, S_2, W, N) vérifiant les conditions précédentes, ainsi qu'un état de départ.

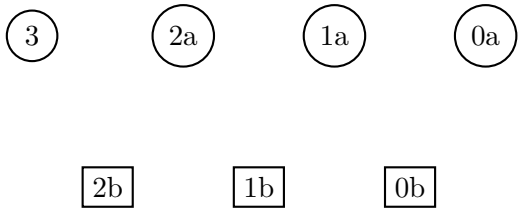
— se donner un *jeu d'accessibilité à deux joueurs à somme nulle* revient à se donner (G, S_1, S_2, W) vérifiant les conditions précédentes, ainsi qu'un état de départ.

Si le graphe des états du jeu n'a pas de cycle, alors les parties sont finies.

Remarque importante
Il est possible que le graphe présenté par l'énoncé soit le graphe des états du jeu. Il est alors très utile dans la pratique de dédoubler le graphe des états du jeu en graphe biparti, de sorte qu'une partie dans le jeu se résume simplement à un chemin dans le graphe biparti, sans qu'on ait à se préoccuper de savoir quel joueur joue. Par exemple, si on a le graphe des états du jeu avec 3 pour état initial :



alors on peut le dédoubler :



Exercice 1 :

1. Écrire les arcs pour un état initial égal à 3.
2. 0 est un état final (il n'a pas de successeur). Convenons que c'est le seul état gagnant et qu'il n'y a pas de partie nulle. Que conseillez-vous au joueur 1 ?

1.2 vocabulaire sur les stratégies

On considère un jeu à deux joueurs J_1 et J_2 modélisé par un graphe biparti sans cycle.

État final : sommet sans successeur. Il y a trois types d'états finals : état gagnant pour J_1, état gagnant pour J_2 et état de partie nulle.

Stratégie : une stratégie consiste à déterminer, pour chaque position, le mouvement suivant à jouer.

Stratégie gagnante pour le joueur J_1 : une stratégie est dite gagnante pour J_1 si toute partie jouée en suivant cette stratégie est gagnante pour J_1.

Position gagnante pour le joueur J_1 : Une position s est dite gagnante pour J_1 lorsqu'**il existe une** stratégie gagnante pour J_1 pour une partie débutant au sommet s.

Un des objectifs est de déterminer par algorithme l'ensemble des sommets qui sont des positions gagnantes pour un joueur.

1.3 les attracteurs

Attracteur (pour le joueur J_1) : On note W l'ensemble des états finals gagnants pour le joueur J_1. On construit par récurrence la suite (A_n), appelée suite d'attracteurs, suivante :

- A_0 est l'ensemble des états finals gagnants pour le joueur J_1 : $A_0 = W$.
- A_n étant donné, A_{n+1} est la réunion de A_n, des sommets contrôlés par J_1 qui possèdent un successeur dans A_n et des sommets contrôlés par J_2 dont tous les successeurs sont dans A_n.

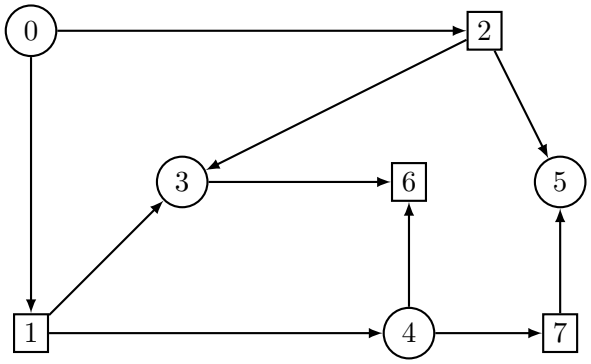
L'attracteur de W est $\mathcal{A} = \bigcup_{j\in\mathbb{N}} A_j$. Comme la suite (A_j) est croissante et incluse dans l'ensemble des sommets, elle est en fait stationnaire et il existe $n_0 \leqslant \text{Card}(S)$ tel que $\mathcal{A} = A_{n_0}$.

Construction de stratégies gagnantes : à partir d'un sommet de A_i, le joueur J_1 possède

une stratégie pour gagner en i coups.

Nous admettons que le premier joueur possède une stratégie gagnante pour tout sommet de $\mathcal{A}$. On définit de manière similaire les attracteurs du joueur J_2.

Exercice 2 : On considère le graphe ci-dessous dans lequel les sommets en forme de cercle sont contrôlés par J_1 et les carrés par J_2. L'état initial est 0. Il y a deux états finals : 5 (gagnant pour J_2 car J_1 ne peut plus jouer) et 6 (gagnant pour J_1 car J_2 ne peut plus jouer).



1. Une stratégie gagnante pour J_1 consiste à jouer :

2. On note $W = \{6\}$.

   $A_0(W) =$
   $A_1(W) =$
   $A_2(W) =$
   $A_3(W) =$
   $A_4(W) =$

   L'ensemble des positions gagnantes pour J_1 est :

   Cela ne veut pas dire que J_1 gagne depuis ces positions, mais qu'il *peut* gagner depuis ces points.

1.4 TP : le jeu de Nim ou jeu des allumettes

Faire le TP correspondant sur CaPytale.

2 Algorithme Min-Max ou Minimax

2.1 d'où vient le terme minimax ?

Dans des jeux simples, à nombre d'états peu élevé, on peut résoudre complètement le jeu et présenter une stratégie gagnante lorsqu'elle existe. Lorsque le nombre d'états ou le nombre de parties est trop élevé, une résolution exhaustive comme nous l'avons faite pour le jeu de Nim avec peu d'allumettes, est cependant impossible. Aux échecs, par exemple, le nombre de parties est environ, selon le décompte de Shannon, de l'ordre de 10^{120}, ou plus en réalité ; au jeu de Go, le nombre de parties est d'environ 10^{600} (source : wikipédia. D'ailleurs, le nombre d'atomes dans l'univers observable actuel est estimé à environ 10^{80}).

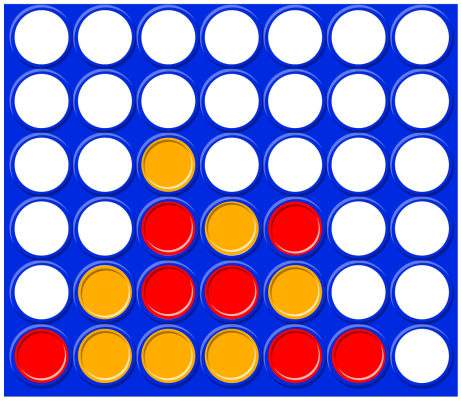
Heuristique

Nous allons utiliser une fonction heuristique H, aussi appelée fonction d'évaluation ou fonction d'utilité. Cette fonction attribue, du point de vue d'un des deux joueurs, un score à chaque sommet. Par exemple, selon le point de vue du joueur J_1,

- plus $H(p)$ est élevé, meilleure est la position p pour J_1
- plus $H(p)$ est faible, meilleure est la position pour J_2.

Aux échecs par exemple, une telle fonction est construite après tâtonnements, avec l'aide de joueurs chevronnés, et prend en compte la différence des pièces entre les deux joueurs, le nombre et la position de cases contrôlées, le nombre de déplacements possibles, etc.

Voici un exemple d'heuristique pour le Puissance 4.



On commence par attribuer à chaque case une valeur, par exemple le nombre d'alignements potentiels de quatre pions lorsqu'on place un pion dans cette case :

3	4	5	7	5	4	3
4	6	8	10	8	6	4
5	8	11	13	11	8	5
5	8	11	13	11	8	5
4	6	8	10	8	6	4
3	4	5	7	5	4	3

Une heuristique possible est égale à la somme des cases occupées positivement pour les pions de J_1 et négativement pour ceux de J_2.
Pour la position donnée ci-contre, J_1 ayant les pions rouges, l'heuristique vaut : $3-4-5-7+5+4-6+8+10-8+11-13+11-11=-1$.

Alternance de max et de min

Grossièrement, en ne prenant pas en compte les états finals, et en se plaçant du point de vue de J_1 au sommet s :

- si J_1 réfléchit uniquement à l'échelle du coup qu'il a à jouer, il choisit de jouer un sommet t réalisant
$$\max_{t \text{ successeur de } s} H(t) \qquad \text{Profondeur 1}$$
- si J_1 anticipe un peu plus, et qu'il réfléchit à son coup et au coup suivant de l'adversaire, il choisit de jouer un sommet t réalisant
$$\max_{t \text{ successeur de } s} \min_{t_2 \text{ successeur de } t} H(t_2) \qquad \text{Profondeur 2}$$
- si J_1 anticipe cette fois sur 3 coups, il choisit de jouer un sommet t réalisant
$$\max_{t \text{ successeur de } s} \min_{t_2 \text{ successeur de } t} \max_{t_3 \text{ successeur de } t_2} H(t_3) \qquad \text{Profondeur 3}$$

On voit apparaître une alternance de phases de maximisation et de minimisation.
Plus la profondeur est grande, meilleure est l'anticipation, mais plus long (lent) est l'algorithme.

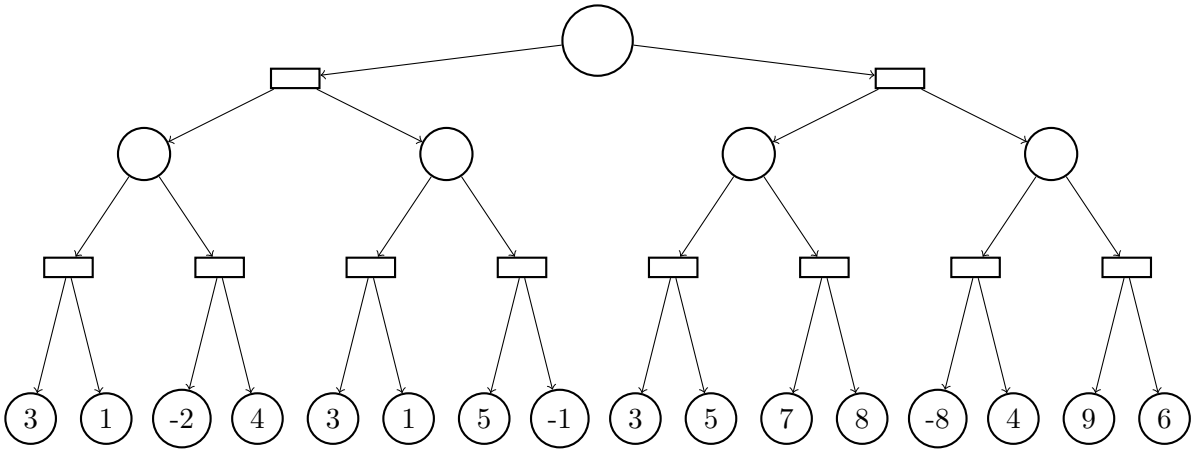
2.2 représentation en arbre de jeu

Pour représenter un jeu, on utilise ici un arbre de jeu. La position initiale est en haut de l'arbre, et est appelée *racine*. Les sommets de l'arbre, appelés *nœuds*, sont des positions du jeu, et les arêtes sont les coups joués par les joueurs.

Exercice 3 : Lecture d'un arbre.

Pour le point de vue de J_1, J_1 maximise l'heuristique et J_2 la minimise.

L'algorithme va procéder à rebours en commençant par étudier les issues « finales pour une certaine profondeur », leur attribuer leur score, puis remonte l'arbre pour attribuer un score à chaque nœud de l'arbre, jusqu'à la racine.
Dans l'arbre ci-dessous, les valeurs de l'heuristique (-5, 2, -3, etc.) sont notées à l'intérieur des nœuds à la place de l'état.



1. Attribuer une note à chaque nœud de l'arbre, en remontant selon le principe :
   — si J_1 a la main sur un nœud, il joue pour atteindre la situation la plus favorable par la suite, donc on attribue au nœud la note maximale des nœuds issus de cette situation,

— si J_2 a la main sur nœud, c'est le contraire.

2. Quel meilleur « gain » le joueur initial peut-il espérer sur ce jeu ?

3. Surligner la partie qui sera jouée si les deux joueurs adoptent la meilleure stratégie en leur faveur.
4. À la suite d'une partie, J_1 gagne 7. Qui s'est trompé et à quel moment ?

5. Après une autre partie, J_1 gagne 5 et il est convaincu d'avoir bien joué. Faut-il lui donner tort ?

2.3 (TP) : programmation d'un duel contre l'ordinateur au jeu du morpion

Faire le TP correspondant sur CaPytale.