

Chapitre 10

Jeux d'accessibilité à deux joueurs

Nous abordons quelques aspects algorithmiques de la théorie des jeux. Nous ne considérerons que des jeux :

- à deux joueurs et à coups asynchrones : le joueur A débute, puis c'est au tour du joueur B, et ainsi de suite,
- à information complète et parfaite : chaque joueur connaît tous les coups que lui et son adversaires peuvent entreprendre et ont préalablement entrepris ainsi que les gains qui en résultent,
- sans hasard,
- à somme nulle : la somme des gains des deux joueurs est nulle : ce que l'un gagne, l'autre le perd. Si l'un gagne, l'autre perd. Il est possible d'aboutir à un match nul, ou à une partie sans fin.

On parle de **jeu d'accessibilité à deux joueurs**.

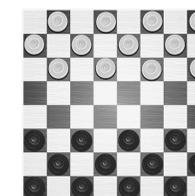
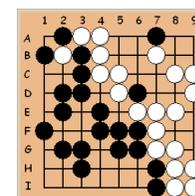
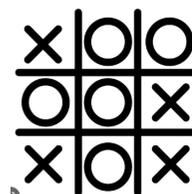
Exemple : une partie de dame, d'échec, de go,... Les jeux de cartes ne sont pas à information complète lorsque un joueur cache sa main, ni sans hasard lorsque les cartes sont distribuées aléatoirement.

Un tel jeu est formalisé à l'aide d'un graphe orienté : appelé graphe du jeu ou arène, dont les sommets sont les états du jeu (les configurations possibles) et les arêtes les coups permettant de passer d'un état à un autre.

Nous nous intéressons à l'obtention de stratégies gagnantes. Nous étudions d'abord dans cette première partie, les jeux dont le graphe est de taille restreinte ; ceci exclut notamment les échecs, etc. L'analyse du graphe peut permettre d'élaborer des stratégies gagnantes.

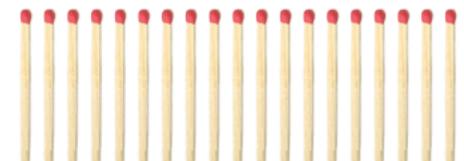
Dans une deuxième partie nous nous intéresserons aux jeux dont l'analyse du graphe n'est pas possible en pratique, à cause de sa taille, et nous verrons des heuristiques pour l'élaboration de stratégies gagnantes.

Ces notions n'ont pas comme seules application les jeux ; mais tous les secteurs de prise de décision, économie, sciences sociales, analyse stratégique, ou encore la biologie évolutionniste, etc. Depuis 1944, 11 prix nobels d'économie ont été décernés pour leur apports à la théorie des jeux.



1 Exemple et formalisation

Prenons l'exemple du jeu de Nim : $N = 20$ allumettes disposées sur une table ; deux joueurs retirent à tour de rôle 1, 2 ou 3 allumettes. Le joueur perdant est celui qui retire la dernière allumette.



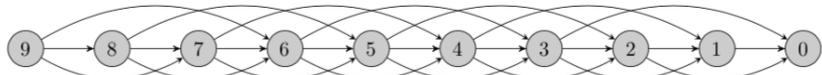
Voilà un exemple de partie où le joueur rouge, celui qui débute, perd :

$$20 \xrightarrow{\text{rouge}} 17 \xrightarrow{\text{bleu}} 16 \xrightarrow{\text{rouge}} 15 \xrightarrow{\text{bleu}} 12 \xrightarrow{\text{rouge}} 10 \xrightarrow{\text{bleu}} 8 \xrightarrow{\text{rouge}} 7 \xrightarrow{\text{bleu}} 4 \xrightarrow{\text{rouge}} 3 \xrightarrow{\text{bleu}} 1 \xrightarrow{\text{rouge}} 0$$

Il n'a pas fait preuve de stratégie. À ce jeu pour le joueur qui débute il y a une stratégie gagnante : une suite de coups qui lui assure la victoire. Laquelle ?

À ce jeu on associe un graphe $G = (S, A)$: les sommets représentent le nombre d'allumettes restantes, et les arêtes les coups permettant de passer d'une configuration à l'autre. Avec $N = 9$ par exemple¹ :

1. Le graphique est extrait du cours de J.Svartz



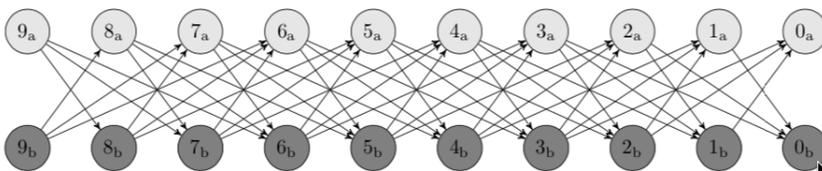
Mais l'information est incomplète : on ne sait pas quelle arête correspond au coup de quel joueur. Pour cette raison on considère un graphe biparti :

Définition 1.1

Un graphe biparti est la donnée d'un graphe $G = (S, A)$ et d'une partition de l'ensemble des sommets S , en $S = S_0 \cup S_1$ avec $S_0 \cap S_1 = \emptyset$ tel que chaque arête de A a son origine dans S_i et son extrémité dans S_{1-i} pour $i = 0, 1$. On le note alors $\mathcal{G} = (G, S_0, S_1)$.

Ainsi un graphe est biparti lorsqu'il y a deux types de sommets, et que chaque arête ne peut relier que des sommets de types différents. En particulier un tel graphe est sans boucle.

Pour le jeu de Nim à 9 allumettes, on duplique les sommets pour préciser à quel joueur, A ou B, est-ce le tour de jouer¹.



Définition 1.2

Un graphe de jeu à deux joueurs, ou arène est un graphe biparti $\mathcal{G} = (G, S_0, S_1)$. On lui associe deux joueurs J_i ($i = 0, 1$) et on dit que S_i est l'ensemble des sommets (ou états) contrôlés par J_i .

On le construit en considérant pour sommets, S_i l'ensemble des configurations du jeu où J_i a la main, et où on relie par une arête $s \in S_i$ à $s' \in S_{1-i}$, lorsqu'un coup du joueur J_i permet de passer de la configuration s à la configuration s' .

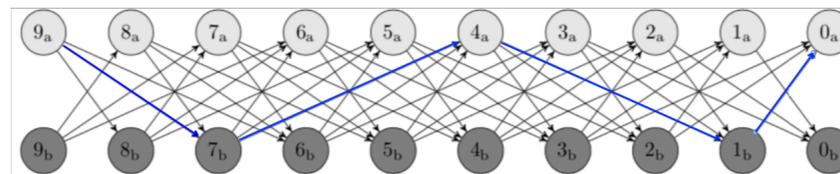
Une partie va alors consister en quel joueur débute, et quelle succession de coups les joueurs enchaînent-ils. C'est donc un chemin débutant en un des deux sommets à gauche (sans prédécesseur) et terminant en un des deux sommets à droite (sans successeurs).

Définition 1.3

Dans un graphe de jeu à deux joueurs $\mathcal{G} = (G, S_0, S_1)$:

- Un sommet (ou état) final est un sommet n'ayant aucun successeur. Il correspond au gain de l'un des deux joueurs, ou à une partie nulle.
- Un chemin est maximal lorsqu'il est infini ou termine en un sommet final.
- Une partie débutant en $s_0 \in S_i$ est un chemin maximal issu du sommet s_0 .
- Une partie partielle débutant en $s_0 \in S_i$ est un chemin issu du sommet s_0 .

Exemple : Dans la partie illustrée ci-dessous le joueur $J_0 = a$ débute, retire deux allumettes, le joueur $J_1 = b$ en retire 3, puis de même pour le joueur a , qui finalement l'emporte¹.



Définition 1.4

Dans un graphe de jeu à deux joueurs $\mathcal{G} = (G, S_0, S_1)$:

- Une condition d'atteignabilité pour le joueur J_i est une partie V_i de l'ensemble des sommets $S = S_0 \cup S_1$; une partie est victorieuse pour J_i si elle passe par un sommet de V_i (pour $i = 0, 1$).

Exemple, $V_0 = \{0_a\}$ est une condition d'atteignabilité pour le joueur a .

Le jeu est pleinement défini dès qu'on connaît son graphe de jeu, une condition d'atteignabilité pour chaque joueur et le sommet de départ.

Parlons maintenant de stratégie :

Définition 1.5

Dans un graphe de jeu à deux joueurs $\mathcal{G} = (G, S_0, S_1)$ avec $G = (S, A)$:

On note S_i ($i = 0, 1$), l'ensemble des sommets dans S_i qui ne sont pas un état final.

- Une stratégie pour le joueur J_i est une application :

$$\sigma : S_i \rightarrow S$$

tel que pour tout $s \in S_i$, $(s, \sigma(s))$ soit un élément de A .

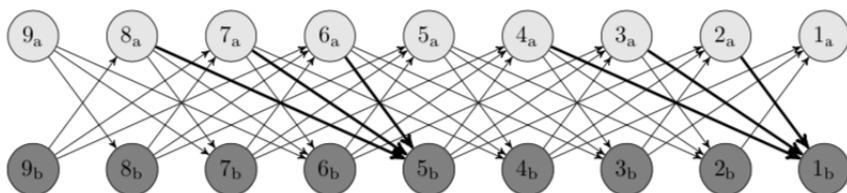
- Lors d'une partie (s_0, s_1, \dots) on dit que la partie suit la stratégie σ si pour tout $s_k \in S_i$ on a $s_{k+1} = \sigma(s_k)$.

– Une stratégie pour le joueur J_i est dite gagnante à partir d'une position s_0 si toute partie débutant en s_0 et qui suit la stratégie aboutit à une condition d'atteignabilité pour J_i , c'est à dire à la victoire de J_i .

Autrement dit, une stratégie est le choix pour chaque état non final contrôlé par le joueur J_i d'un coup à jouer, et une partie qui suit cette stratégie est une partie où le joueur J_i s'impose à chaque coup de jouer ce coup. Elle est gagnante à partir du sommet initial si elle assure la victoire.

Remarque. Il s'agit de ce qu'on appelle une stratégie sans mémoire : les coups à jouer ne dépendent que de l'état actuel du jeu et non des coups précédents. Au jeu d'échec la partie est déclarée nulle si une même configuration de jeu se répète trois fois ; une stratégie sans mémoire ne présente plus d'intérêt.

Exemple. La stratégie au jeu de Nim consistant à laisser à l'adversaire un nombre d'allumette congru à 1 modulo 4 s'avère gagnante depuis tous les états de départ où le joueur n'a pas $4n + 1$ allumettes¹.



En gras les images par une stratégie gagnante pour le joueur a au départ d'un état avec un nombre d'allumette $\not\equiv 1[4]$; les valeurs de la stratégie aux sommets $9a$ et $5a$ sont sans importance et définissent tous une stratégie gagnante pour a au départ de $8a, 7a, 6a, 4a, 3a, 2a$.

Définition 1.6

Un sommet $s \in S_i$ est une position gagnante pour le joueur J_i , si il existe une stratégie gagnante pour J_i à partir de la position s ($i = 0, 1$).

Exemple. Les sommets $8a, 7a, 6a, 4a, 3a, 2a$ sont des positions gagnantes pour le joueur a ; les sommets $9a$ et $5a$ ne le sont pas (quoi qu'il joue il peut perdre, si le joueur adverse suit la stratégie gagnante évoquée, puisque par exemple au départ de $9a$, on ne peut pas jouer un coup qui laisse $4n + 1$ allumettes à l'adversaire).

2 Résolution du jeu : attracteurs

Dans un jeu d'accessibilité, résoudre le jeu consiste à déterminer les positions gagnantes et pour chacune une stratégie gagnante. À cette fin on utilise la notion d'attracteurs.

Pour le joueur J_0 , appelons V_0 l'ensemble des états finaux correspondant à une victoire du joueur J_0 . À une position donnée, le joueur J_0 est à un coup de l'emporter, soit si c'est à lui de jouer et qu'il peut choisir un coup pour atteindre V_0 , soit si c'est au tour de son adversaire de jouer et que tous ses coups possibles le conduiront à V_0 . En particulier sa position est une position gagnante.

Définition 2.1

Dans un graphe de jeu à deux joueurs $\mathcal{G} = (G, S_0, S_1)$ avec $G = (S, A)$:

- On note V_0 l'ensemble des sommets finaux assurant la victoire du joueur J_0 .
- On définit alors une suite $(V_n)_n$ de sommets (états du jeu) par :

$$V_{n+1} = V_n \cup \underbrace{\left\{ s \in S_0 \mid \exists (s, s') \in A, s' \in V_n \right\}}_{V_n \text{ peut être atteint en un coup par } J_0} \cup \underbrace{\left\{ s \in S_1 \mid \forall (s, s') \in A, s' \in V_n \right\}}_{\text{en un coup } J_1 \text{ ne peut qu'atteindre } V_n}$$

- Le bassin d'attraction de V_0 est l'ensemble des sommets :

$$Attr(V_0) = \bigcup_{n=0}^{+\infty} V_n$$

Ses sommets sont appelés des attracteurs de V_0 .

Informellement, les éléments de V_1 sont positions gagnantes en au plus un coup, de V_2 en au plus deux coups, etc. Comme le montre le résultat fondamental suivant :

Théorème 2.1

Le bassin attracteur de V_0 est exactement l'ensemble des positions gagnantes pour le joueur J_0 .

Preuve. Notons pour un sommet $s \in Attr(V_0)$ son rang : $rang(s) = \min \{ k \in \mathbb{N} \mid s \in V_k \}$ et montrons par récurrence forte sur le rang n , que tout sommet $s \in Attr(V_0)$ est une position gagnante pour le joueur J_0 .

(I) Pour $n = 0$ c'est clair : tout sommet de V_0 est une position gagnante pour J_0 .

(H) Soit $n \in \mathbb{N}$; supposons que pour tout $k \leq n$, tout sommet de $Attr(V_0)$ de rang k soit une position gagnante. Soit $s \in Attr(V_0)$ un sommet de rang $n + 1$. Par définition :

– Si le sommet s est contrôlé par J_0 , alors il existe une arête $(s, s') \in A$ avec $s' \in V_n$; puisque $s' \in Attr(V_0)$ est de rang $\leq n$, par (HR) c'est une position gagnante, c'est donc aussi le cas de s : une stratégie gagnante pour J_0 existe au départ de s .

– Si le sommet s est contrôlé par J_1 , alors pour toute arête $(s, s') \in A$, nécessairement $s' \in V_n$; puisque $s' \in Attr(V_0)$ est de rang $\leq n$, par (HR) c'est une position gagnante, c'est donc aussi le cas de s : une stratégie gagnante pour J_0 existe au départ de s qu'aucun coup de J_1 ne pourrait empêcher.

Ceci conclut la récurrence; il reste à montrer qu'aucun sommet hors de $Attr(V_0)$ n'est une position gagnante pour J_0 . Soit un tel sommet s : selon s'il est contrôlé par J_0 ou J_1 , aucun coup de J_0 ne lui permettra

d'atteindre $Attr(V_0)$, et au moins un coup de J_1 lui permettra de rester hors de $Attr(V_0)$. Par une récurrence immédiate, J_1 pourra établir une stratégie pour rester toujours hors de $Attr(V_0)$, et donc pour que la partie ne puisse aboutir en V_0 c'est à dire pour que J_0 ne puisse l'emporter. ■

Remarque. Si ni un match nul ni une partie infinie ne sont possibles, l'argument précédent montre que le bassin d'attraction des états finaux victorieux pour le second joueur $Attr(V_1)$ est le complémentaire dans S de $Attr(V_0)$. Et donc tout état est une position gagnante pour l'un des deux joueurs.

Le calcul des positions gagnantes passe donc par le calcul du bassin attracteur ; il s'obtient par un algorithme de parcours en profondeur sur le graphe à partir des sommets de V_0 mais en suivant les arêtes en sens inverse : à partir de V_k , on construit V_{k+1} en considérant tous les sommets de S_0 ayant un successeur dans V_k ainsi que tous les sommets de S_1 dont tous les successeurs sont dans V_k . Pour traiter ce dernier cas des sommets de S_1 , il sera nécessaire de calculer leur degré sortant.

• Algorithme de calcul du bassin d'attraction :

```

Données : un graphe biparti  $G$ , une partition de ses sommets  $S_0, S_1$ ,
           un ensemble  $V_0$  de sommets.
Sortie  : le bassin d'attraction de  $V_0$ 
Attr  $\leftarrow \emptyset$ 
Pour tout  $s$  sommet de  $G$ :
     $d_s \leftarrow$  degré sortant de  $s$  dans  $G$ 
# Partie récursive (parcours en profondeur en sens inverse)
Fonction  $\text{parcours}(s)$  :
    Si  $s \notin Attr$  Alors :
         $Attr \leftarrow Attr \cup \{s\}$ 
        Pour tout  $s'$  prédécesseur de  $s$  dans  $G$  :
             $d_{s'} \leftarrow d_{s'} - 1$ 
            Si  $s' \in S_0$  ou  $d_{s'} = 0$  Alors :
                 $\text{parcours}(s')$ 
# Boucle principale :
Pour tout  $s \in V_0$  :
     $\text{parcours}(s)$ 
Retourner  $Attr$ 

```

• Terminaison de l'algorithme :

En dehors de la fonction parcours la terminaison est évidente. La fonction parcours exécute un parcours en profondeur (en sens inverse) et termine à cause du test initial Si $s \notin Attr$ qui assure que chaque sommet sera traité au plus une fois.

• Correction de l'algorithme :

On montre deux implications.

Commençons par montrer que la propriété "Tout élément de $Attr$ est un sommet attracteur" est un invariant de l'algorithme.

- L'insertion d'un élément dans $Attr$ ne s'effectue qu'en début d'appel de $\text{parcours}(s)$.
- Initialement c'est clair puisque $Attr$ est vide.
- Lorsque $\text{parcours}(s)$ est lancé depuis la boucle principale, la propriété est vraie après l'insertion de s dans $Attr$ puisque $s \in V_0$ est bien un sommet attracteur.
- Sinon l'insertion de s' a lieu à l'appel de $\text{parcours}(s')$ au sein de $\text{parcours}(s)$ où s est un sommet attracteur.
 - Si s' est dans S_0 alors (s', s) est une arête de G et donc par définition, s' est un sommet attracteur.
 - Si s' n'est pas dans S_0 , alors $d_{s'}$ vaut 0 ce qui signifie que tous les successeurs de s' sont des éléments déjà visités de $Attr$ et donc des sommets attracteurs. Ainsi s' est aussi un sommet attracteur.

Montrons maintenant par l'absurde que tout sommet attracteur se trouve dans $Attr$ à la fin de l'algorithme. Supposons qu'il existe des sommets attracteurs qui ne soient pas dans $Attr$, et soit s un tel sommet de rang maximal (cf. preuve du théorème).

- Le rang de s ne peut être 0.
 - Si $s \in S_0$, alors par hypothèse il existe un successeur s' de s qui est un sommet attracteur de rang strictement inférieurs, et qui est donc dans $Attr$. Mais dans ce cas durant l'appel de $\text{parcours}(s')$ où s' est ajouté à $Attr$, $\text{parcours}(s)$ aurait du être appelé et s ajouté à $Attr$. Contradiction.
 - Si $s \notin S_0$, alors par hypothèses tous les successeurs de s sont des sommets attracteurs de rang strictement inférieurs, et sont donc tous dans $Attr$; ils sont en nombre la valeur initiale de d_s qui a été décrémenté à chaque insertion dans $Attr$. Si s' est le dernier de ces successeurs à avoir été ajouté à $Attr$, durant $\text{parcours}(s')$, durant la boucle qui énumère les prédécesseur de s' , la variable d_s est passée à 0, et $\text{parcours}(s)$ a été appelée, avec pour première effet l'ajout de s dans $Attr$. Contradiction.
- Ceci prouve la correction de l'algorithme : l'ensemble $Attr$ renvoyée est constituée de tous les sommets attracteurs de V_0 .

• Complexité de l'algorithme :

En implémentant le graphe à l'aide d'une liste d'adjacence, les sommets à l'aide d'entiers entre 0 et $\#S$, et S_0 par exemple par une liste de booléens de longueur $\#S$ ou $S_0[i]$ caractérise le fait que le sommet i soit dans S_0 , la complexité de l'algorithme est en $O(\#S + \#A)$.

• **Programme Python :**

On se donne le graphe à l'aide d'une matrice d'adjacence :

```
# Le graphe G est donné par une liste d'adjacence
# Les sommets par des entiers entre 0 et len(G)-1
# V0 par une liste de sommets
# S0 par une liste de booléens

def grapheTranspose(G):
    n = len(G)
    Gt = [[ ] for _ in range(n)]
    for i in range(n):
        for j in G[i]:
            Gt[j].append(i)
    return Gt

def bassinAttracteur(G, S0, V0):
    n = len(G)
    Attr = [False] * n
    dG = [ len(v) for v in G ] # liste des degrés sortants
    Gt = grapheTranspose(G)
    # Parcours en profondeur récursif :
    def parcours(s):
        if not Attr[s]:
            Attr[s] = True
            for s1 in Gt[s]:
                dG[s1] = dG[s1] - 1
                if S0[s1] or dG[s1] == 0:
                    parcours(s1)
    for s in V0:
        parcours(s)
    return Attr
```

Pour calculer simultanément une stratégie gagnante, il suffit de poser à l'appel de `parcours(s1)` durant `parcours(s)`, $\sigma(s) = s1$:

```
def bassinAttracteur(G, S0, V0):
    ...
    ...
    Strategie = [False] * n # !!!
    def parcours(s):
```

```
...
...
        if S0[s1] or dG[s1] == 0:
            Strategie[s] = s1 # !!!
            parcours(s1)
...
...
return Attr, Strategie # !!!
```