TP : le jeu de Chomp

Nous allons nous intéresser à l'aide de l'exemple du jeu de Chomp, à l'étude théorique et algorithmique de jeux à deux joueurs antagonistes jouant alternativement, joueurs que l'on appellera J_1 et J_2.
Les jeux qui nous intéressent sont à information totale : à tout instant d'une partie, chacun des joueurs a une vision complète de l'état du jeu. Les échecs, les dames, le go font partie des jeux à information totale.

1 Règles du jeu de Chomp

Le jeu de Chomp se joue à l'aide d'une tablette de chocolat rectangulaire dont le coin supérieur gauche est empoisonné : chaque joueur choisit à tour de rôle un carré et le mange, ainsi que tous les morceaux situés à la droite et en dessous du carré choisi. Bien évidemment, le joueur qui n'a plus d'autre choix que de manger le carré empoisonné a perdu.

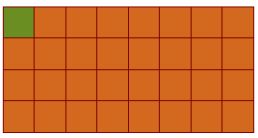


Figure 1 – La configuration initiale du jeu de Chomp.

On trouvera figure 2 un exemple de partie perdue par Adam (J_1), qui a commencé à jouer en premier.

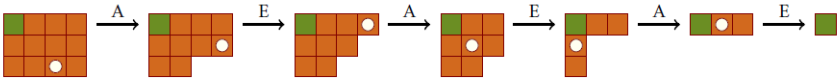


Figure 2 – Un exemple de partie du jeu de Chomp.

2 Représentation du jeu par un graphe

À ce type de jeu est associé un graphe orienté (S ;A) appelé arène. Chaque sommet de S représente une configuration du jeu (une position), l'une d'entre elles étant la position de départ. Une arête a = (s1 ; s2) 2 A reliant deux sommets indique la possibilité pour un joueur de passer de la position s1 à la position s2 en un coup. Par exemple, l'arène associée au jeu de Chomp pour une tablette (2 ; 3) est représentée figure 3. Pour visualiser une partie de Chomp, il suffit d'imaginer un jeton initialement posé sur la position initiale s0. À tour de rôle, chaque joueur le déplace le long d'une arête issue de la position courante s et le pose sur un successeur de s. Une partie est donc un chemin d'origine s0 dans l'arène. Ce jeu fait partie des jeux d'accessibilité : le graphe associé ne comporte pas de cycle (ce qui assure que toute partie est finie), et il est déterminé par un ensemble de positions (les cibles) qui sont sans successeurs. Ainsi, dans le jeu de Chomp le noeud du graphe associé au seul carré empoisonné est la seule cible du jeu, et l'atteindre signifie la fin de la partie (et la victoire pour celui qui l'atteint).

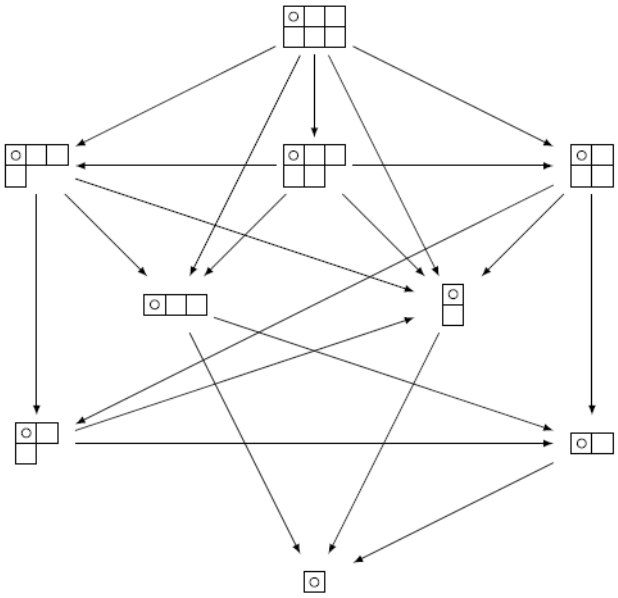


FIGURE 3 – L'arène associée au jeu de Chomp (2, 3).

Une fois fixé le joueur qui commence (Adam par exemple), on peut, en doublant chacun des sommets qu'on indexera par le nom du joueur, créer un nouveau graphe dont les sommets seront partitionnés en deux sous-ensembles $S = S_a \cup S_e$ avec $S_a \cap S_e = \emptyset$ où S_i est lensemble des positions à partir desquelles le joueur i jouera. Un tel graphe est dit biparti (illustration figure 4).

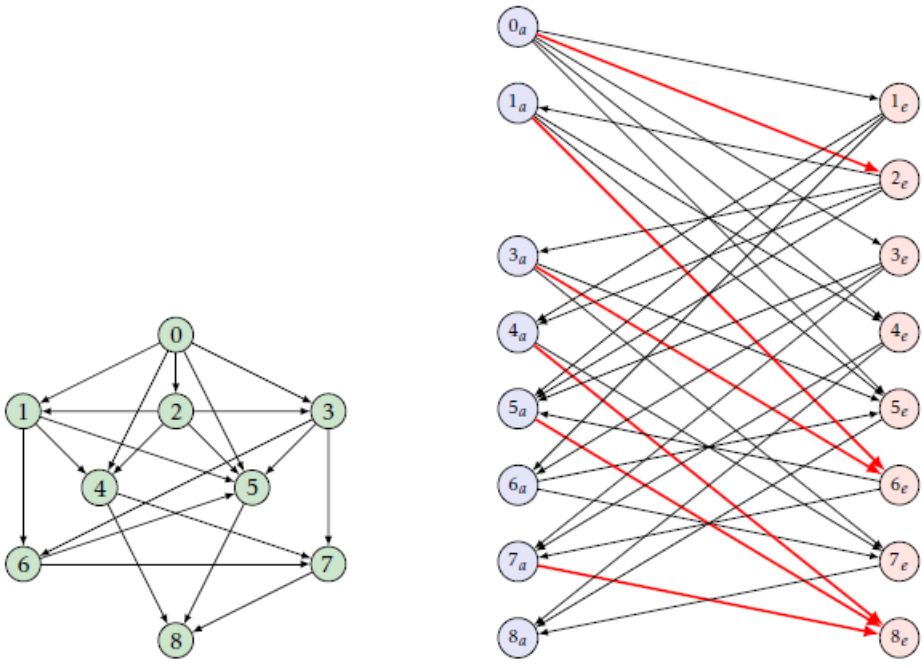


FIGURE 4 – Graphe et graphe biparti associés au jeu de Chomp (2, 3).

Attention, pour notre TP, nous raisonnerons essentiellement à partir du graphe des positions, et non pas du graphe biparti.

3 Quelques définitions

Définition (Graphe biparti) –
Un graphe (orienté ou non-orienté) est biparti si l'on peut diviser l'ensemble de ses sommets en deux sous-ensembles disjoints S_1 et S_2 tels que toute arête du graphe a une extrémité dans S_1 et l'autre dans S_2.

Les sommets de S_1 (resp. S_2) seront les états contrôlés par J_1 (resp. J_2)

Définition (Jeu d'accessibilité) –
Un jeu d'accessibilité à deux joueurs J_1 et J_2 est défini par :

- un graphe (fini) orienté $\mathcal{G} = (S, A)$, biparti selon la partition $S = S_1 \cup S_2$ (états contrôlés par J_1 et J_2 respectivement), appelé *arène* ;
- un état de départ $s_0 \in S$;
- une partition de l'ensemble F des états finals, sous la forme $F = G_1 \cup G_2 \cup N$ (états finals gagnants pour J_1, gagnants pour J_2, et nuls respectivement).

Une partie est un chemin de $\mathcal{G}$ qui part de s_0 et qui soit termine en un état de F (état final), soit est infini (cas où le graphe possède un cycle)

Définition (Stratégie) –
Soit $\mathcal{G} = (S_1 \cup S_2), A)$ une arène et $i \in \{1, 2\}$. Une stratégie (sans mémoire) pour J_i est une fonction $f : S_i \mapsto S_{3-i}$ telle que $(s, f(s)) \in A$ pour tout état $s \in S_i$ qui n'est pas un état final de $\mathcal{G}$.

Pour J_i, suivre la stratégie f consiste, lorsqu'il doit jouer à partir de l'état (non final) $s \in S_i$, à aller systématiquement à l'état $f(s)$. Autrement-dit, le joueur fixe avant le début de la partie les coups qu'il va jouer, pour chaque coup possible de son adversaire.

Par exemple, si le joueur J_1 commence, il peut décider de jouer $0 \to 3$.
Puis, pour chacun des coups de J_2 : $3 \to 5$, $3 \to 6$, $3 \to 7$, J_1 choisit son coup.
Par exemple : $5 \to 8$, $6 \to 7$, $7 \to 8$.
Il n'y alors plus de coups envisageables car le seul coup restant possible pour J_2 après $6 \to 7$ est $7 \to 8$ qui termine la partie.
Par conséquent, la stratégie choisie ci-dessus est définie par :

$$0 \to 3 \quad 5 \to 8 \quad 6 \to 7 \quad 7 \to 8$$

En général, il n'est pas nécessaire qu'une stratégie pour J_i soit définie sur S_i tout entier. Pour qu'elle soit applicable, il suffit que son ensemble de définition D_f (qui est inclus dans S_i) vérifie :

- si $s_0 \in S_i$, $s_0 \in D_f$.
- si le successeur t d'un état s appartenant à l'image de f, n'est pas un état final, $t \in D_f$.

Définition (Stratégie gagnante) –
Soit $\mathcal{G} = (S, A)$ une arène dont l'ensemble des états finals se décompose en $G_1 \cup G_2 \cup N$. Une stratégie est gagnante pour J_1 si toute partie $\pi = (s_0, s_1, s_2, \cdots)$ vérifiant $s_{2n+1} = f(s_{2n})$ pour tout n est finie et termine en un état de G_1 (état gagnant pour G_1).

Sur le graphe biparti de la figure 4, on peut déterminer une stratégie gagnante pour le joueur Adam (J_1) qui commence à jouer :
il doit commencer par jouer $0 \to 2$. Puis,

— si Eve (J_2) joue son prochain coup sur 1 ou 3, il peut ensuite jouer en 6. Eve ne peut alors que suivre sur 5 ou 7 et Adam peut conclure en 8 ;

— si Eve joue son coup suivant sur 4 ou 5, Adam peut terminer sur 8.

Définition (Position gagnante) –
Une position s est dite gagnante lorsqu'il existe une stratégie gagnante pour une partie débutant en s.

Par conséquent, le joueur J_i dispose d'une stratégie gagnante si et seulement si s_0 est une position gagnante. Par exemple, dans le jeu de Chomp(2,3), les positions 0, 1, 3, 4, 5, 7 sont gagnantes, les positions 2, 6, 8 sont perdantes.

Définition (Attracteur) –
Pour un ensemble d'états $F \subset S$ de $\mathcal{G} = (S, A)$, on appelle attracteur de F pour le joueur J_i, l'ensemble des positions à partir desquelles le joueur i possède une stratégie qui lui assure d'arriver à un état de F en un nombre de coups finis.

L'attracteur de l'ensemble des états finals gagnants est l'ensemble des positions gagnantes.

Plutôt que d'utiliser la notion d'attracteur, qui nécessite de raisonner sur le graphe biparti, nous verrons dans ce TP la notion de noyau pour calculer l'ensemble des positions gagnantes.

Définition (Noyau) –
Un sous-ensemble S' de sommets de (S, A) est dit *stable*, si tout sommet de S' n'a aucun successeur dans S'. Un sous-ensemble S' de sommets est dit *absorbant* si tout sommet n'appartenant pas à S' possède au moins un successeur dans S'.
Un sous-ensemble stable et absorbant est appelé un noyau du graphe (S, A).

Pour calculer le noyau de (S, A), on dispose de l'algorithme suivant :

1. chercher un sommet s de (S, A) sans successeur et ajouter s dans le noyau ;
2. supprimer de (S, A), s et ses prédécesseurs ;
3. recommencer tant qu'il reste des sommets dans (S, A)

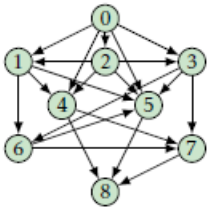
Proposition –
Si $\mathcal{G}$ est acyclique et tous les états finals sont gagnants (pas de partie nulle), alors soit il existe une stratégie gagnante pour J_1, soit il existe une stratégie gagnante pour J_2.

4 Calcul des positions gagnantes dans le jeu de Chomp

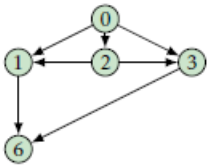
4.1 Principe

Nous utilisons le principe du calcul du noyau.
Celui-ci va nous permettre d'accéder aux positions gagnantes (ou perdantes) :

- 8 est une position perdante (pour le joueur qui à la main en 8) ;
- 4,5,7 sont les prédécesseurs de 8 donc 4,5,7 sont des positions gagnantes ;
- les voisins de 6 sont tous des positions gagnantes (5 et 7) donc 6 est une position perdante ;
- 1 est un prédécesseur d'une position perdante (6) donc 1 est une position gagnante ;
- les voisins de 2 sont tous des positions gagnantes (1,3,4,5) donc 2 est une position perdante
- 0 est une position gagnante car un prédécesseur d'une position perdante 2.



8 n'a pas de successeur ; il appartient au noyau et on le supprime ainsi que tous ses prédécesseurs.



6 n'a pas de successeur ; il appartient au noyau et on le supprime ainsi que tous ses prédécesseurs.

2 n'a pas de successeur ; il appartient au noyau et on le supprime ainsi que tous ses prédécesseurs.

Le noyau est donc (2, 6, 8).

Dans le cas du jeu de Chomp les éléments du noyau sont les positions perdantes et les autres sommets les positions gagnantes (car chaque prédécesseur d'une position perdante est une position gagnante). La stratégie gagnante consiste, pour chaque sommet qui n'est pas dans le noyau, à jouer un coup qui s'y ramène.

4.2 Représentation du graphe du jeu de Chomp

Le graphe du jeu de Chomp est représenté en machine par la donnée d'un dictionnaire d dont les clefs sont les sommets et les valeurs les successeurs des clefs (sous forme de liste). Par exemple, le graphe du jeu de Chomp (2,3) est représenté en mémoire par le dictionnaire :

```
d = { 0: [1, 2, 3, 4, 5], 1: [4, 5, 6], 2: [1, 3, 4, 5],
3: [5, 6, 7], 4: [7, 8], 5: [8], 6: [5, 7], 7: [8], 8: []}
```

4.3 Implémentation

1. Rédiger une fonction `sansSuccesseur(d)` qui renvoie un sommet sans successeur.
2. Rédiger une fonction `predecesseurs(d,j)` qui renvoie la liste de tous les prédécesseurs du sommet `j`.
3. Rédiger une fonction `supprimeSommet(d,j)` qui supprime le sommet du graphe (cette fonction modifiera le dictionnaire `d`.

   Indication : on rappelle que la commande `del d[j]` supprime la clé j (et la valeur associée) d'un dictionnaire d.

   On pourra écrire une fonction intermédiaire `supprimeElement(L,j)` qui renvoie une liste constituée des éléments de liste `L`, privée de l'élément `j` s'il appartient à `L`.
4. En déduire une fonction `noyau(d)` qui renvoie la liste des sommets constituant le noyau de (S, A) représenté par d (c'est-à-dire les positions perdantes du jeu de Chomp).

   Attention : on pourra commencer par créer une copie du dictionnaire `d` à l'aide de la commande `d2 = d.copy()` pour ne pas modifier `d` lors de l'exécution de la fonction `noyau`.
5. Préciser la complexité de la fonction `noyau(d)` en fonction de $n = \text{len}(d)$.

   Remarque : sachant que le jeu Chomp(p,q) possède $n = \binom{p+q}{p} - 1$ *sommets, on conçoit que le calcul du noyau se révèle d'un coût rédhibitoire dès lors que* p *et* q *sont grands. Par exemple, pour* $p = q$, $n \sim \binom{2p}{p} \sim \frac{4^p}{\sqrt{\pi p}}$ *donc la complexité croît exponentiellement avec* p.

6. Ecrire une fonction (récursive) `strategie(d)` qui renvoie une stratégie gagnante (pour le joueur qui commence à jouer).

   Une stratégie sera représentée par un dictionnaire dont les clés sont des sommets du graphe et à chacun de ses sommets, on associe le sommet correspondant au coup "gagnant", c'est-à-dire un sommet du noyau de `d`.

   Par exemple, pour le jeu Chomp(2,3), une stratégie gagnante est représentée par :

   `strat = { 0:2 , 1:6 , 3:6 , 4:8 , 5:8 , 7:8 }`.

   Pour cette question, on pourra compléter la fonction suivante :

```
def strategie(d):
    strat={}
    Lnoy = noyau(d)
    def completerStrat(s):  # fonction qui va construire le dico strat
        if len(d[s])>0:
            # à compléter pour definir strat[s]

            # pour chacun des voisins de strat[s]
            for j in d[strat[s]]:
                # appel récursif à compléter
    completerStrat(0)
    return strat
```

5 Fonction de Sprague-Grundy

On considère un graphe orienté acyclique (S, A) associé à un jeu impartial dans lequel le perdant est celui qui ne peut plus jouer. On définit le nimber $n(s)$ d'un sommet $s \in S$ de la façon suivante :

— si s est une position gagnante (une position sans successeur), $n(s) = 0$;

— sinon, $n(s)$ est le plus petit entier positif ou nul n'apparaissant pas dans la liste des nimbers de ses successeurs.

On peut démontrer que l'ensemble des sommets $s \in S$ vérifiant $n(s) = 0$ est le noyau du graphe :
Posons $S' = \{s \in S \mid n(s) = 0\}$. Par définition S' est stable car un sommet de S' ne peut avoir parmi ses successeurs un autre sommet de nimber nul.
De plus pour $s \in S \setminus S'$, $n(s) > 0$ donc s possède un successeur de nimber nul, autrement dit appartenant à S'. S' est donc absorbant.

1. Déterminer le nimber de chacun des sommets du graphe du jeu Chomp(2,3).

   Le but des questions suivantes est de construire un dictionnaire `n` dont les clés sont les sommets et les valeurs les nimbers de ces sommets.

2. Rédiger une fonction `sansNimber(d,n)` qui renvoie un sommet `s` ne possèdant pas encore de nimber mais dont tous les successeurs en possèdent. Cette fonction renverra `None` dans le cas où un tel sommet n'existe pas.

3. En déduire une fonction `nimber(d)` qui renvoie le dictionnaire des nimbers des différents sommets composant ce graphe.