THÉORIE DES JEUX

> Si deux ont proposé entre eux, de dire chacun l'un après l'autre alternativement un nombre à plaisir, qui toutefois ne surpasse pas un certain nombre précis, pour voir ajoutant ensemble les nombres qu'ils diront qui arrivera plutôt à quelque nombre prescrit ; faire si bien quon arrive toujours le premier au nombre destiné.
>
> Claude-Gaspar Bachet de Méziriac, 1612
> [**bachet_problemes_1612**]

À la fin de ce chapitre, je sais :

- ☞ expliquer l'intérêt de la théorie des jeux
- ☞ expliquer le concept de jeu d'accessibilité
- ☞ coder le calcul des attracteurs
- ☞ expliquer la notion d'heuristique
- ☞ appliquer les algorithmes A* et minimax

La théorie des jeux a été initié par John Von Neumann pendant et après la seconde guerre mondiale. Dans un ouvrage resté célèbre[**von_neumann_theory_1944**], de nombreux problèmes très généraux sont abordés sous la perspective du jeu et de l'économie. Tout comme les algorithmes d'IA développés aujourd'hui, cette théorie s'inscrit dans l'objectif d'aider à la décision lorsque l'environnement est incertain, c'est à dire complexe et imprévisible. Elle fait intervenir des joueurs considérés comme des individus rationnels, des règles et des contextes d'évolution du jeu.

Le programme de classe préparatoire n'aborde que les jeux d'accessibilité à deux joueurs que l'on peut modéliser avec un graphe orienté biparti. Néanmoins, cela permet de lever le voile sur une théorie puissante et fascinante.

A Introduction à la théorie des jeux

■ **Définition 1 — Jeu.** Dans le cadre de cette théorie, on considère qu'un jeu est une activité humaine définie dans le cadre d'un contexte et dont les participants doivent suivre les règles énoncées et faire des choix pour gagner en s'opposant ou résoudre un problème ensemble. Cette activité nécessite des compétences intellectuelles, des savoirs et incorpore le hasard.

Les jeux ainsi définis englobent donc la plupart des activités humaines : l'économie, la guerre, l'étude du vivant ou même la physique peuvent être le cadre de jeux qui servent alors de modèles pour découvrir, établir des stratégies ou simuler une réalité.

■ **Définition 2 — Jeux coopératifs.** Un jeu coopératif permet la construction de coalitions entre joueurs. Cela suppose une concertation sur la stratégie à adopter et un engagement à coopérer.

■ **Définition 3 — Jeu à somme nulle.** Les jeux à somme nulle sont des jeux à deux joueurs pour lesquels les gains de l'un sont strictement les pertes de l'autre. Si on utilise une fonction de gain pour évaluer les perspectives de gain de chaque joueur, alors la somme des deux fonctions de gain est nulle.

■ **Exemple 1 — Jouer à somme nulle.** Parmi les jeux les plus connus à somme nulle, on trouve :

- les échecs,
- les jeux de carte comme la belote, le tarot ou le poker,
- shi-fu-mi.

R La plupart des situations de la vie quotidienne engendre des jeux à somme non nulle. Par exemple, le commerce est un jeu à somme non nulle plutôt positive : un marchand de voiture est gagnant lors qu'il vend une voiture à un client et son gain n'est pas égal à l'inverse du crédit qu'a souscrit l'acheteur... Néanmoins, cela ne signifie pas qu'ils ont perdu pour autant, les situations commerciales peuvent être gagnant-gagnant : si vous avez faim, vous serez content d'acheter de la nourriture qu'un marchant voudra bien vous vendre.

■ **Définition 4 — Dilemme du prisonnier.** Le dilemme du prisonnier est un exemple fondamental [a] de la théorie des jeux. Il a été formalisé par Tucker en 1950 [**tweedale_william_1993**] pour pointer une insuffisance de la théorie des jeux de l'époque : deux individus rationnels ne coopèrent pas nécessairement [b]. Le principe est le suivant :

Deux membres d'un même gang criminel sont arrêtés et emprisonnés. Chaque prisonnier est mis à l'isolement : il ne peut pas communiquer avec l'autre. La police ne dispose pas de suffisamment de preuves pour les accuser formellement tous les deux et il est envisagé de les condamner à un an de prison tous les deux pour des charges moindre. Pour l'instant les deux prisonniers gardent le silence.

Néanmoins, la police propose à chaque prisonnier A et B un marché diabolique. En voici

les termes :

1. Si A et B se dénoncent mutuellement, il seront condamnés à deux ans de prisons.
2. Si A trahit B et que B demeure silencieux, A sera libéré et B sera condamné à trois ans.
3. Symétriquement, si A demeure silencieux et que B le dénonce, alors A fera trois ans et B sera libéré.
4. Enfin, si A et B demeurent silencieux, les deux feront un an de prison.

a. un paradigme

b. On trouve ici [**arte_dilemme_2021**] une fabuleuse introduction à ce dilemme dans la série Voyages au pays des maths d'Arte. À regarder absolument !

R Le dilemme du prisonnier illustre bien des situations (guerre commerciale par exemple) dans lesquelles les acteurs peuvent agir rationnellement, ne pas coopérer spontanément et perdre simultanément. L'incitation à tricher est naturellement au cœur du dilemme.

La répétition du jeu peut cependant amener à considérer d'autres stratégies : chaque joueur peut adapter son comportement par rapport à l'expérience passée et choisir de coopérer ou au contraire de se venger. Lorsque l'incitation à tricher est moins forte que les représailles potentielles, la coopération peut alors s'imposer et le jeu peut atteindre un équilibre de Nash.

■ **Définition 5 — Jeu séquentiel.** Un jeu séquentiel est un jeu au cours duquel les joueurs décident de leur stratégie les uns après les autres et peuvent donc tenir compte des actions des joueurs précédents.

■ **Définition 6 — Jeu à information parfaite.** Un jeu est à information parfaite si chaque joueur est parfaitement informé des actions passées des autres joueurs avant de prendre sa décision : aucune action du jeu n'a été cachée. On se rappelle de tous les coups joués précédemment. Un jeu à information parfaite est un jeu séquentiel.

■ **Définition 7 — Jeu à information complète.** Un jeu à information est à information complète si tous les joueurs ont une connaissance totale des données du jeu : règles, pièces, actions possibles, fonction de gain, objectifs des autres joueurs.

R Les jeux à information incomplète sont appelés jeux bayésiens. Dans ce cas, les joueurs n'ont pas une connaissance commune du jeu : chacun n'a qu'une vision partielle des données du jeu.

■ **Exemple 2 — Jouer à information (in)complète et (im)parfaite.** Aux échecs, s'il s'agit d'une partie d'échec classique, les joueurs évoluent dans un contexte d'information parfaite : chaque joueur a pu voir tous les coups joués précédemment au cours de la partie. De plus, les règles sont connues, toutes les pièces sont toutes visibles, le chronomètre aussi :

alors l'information est complète également. Par contre, si une partie est pris en cours de route et que le joueur n'a pas connaissance des coups passés, l'information est imparfaite.

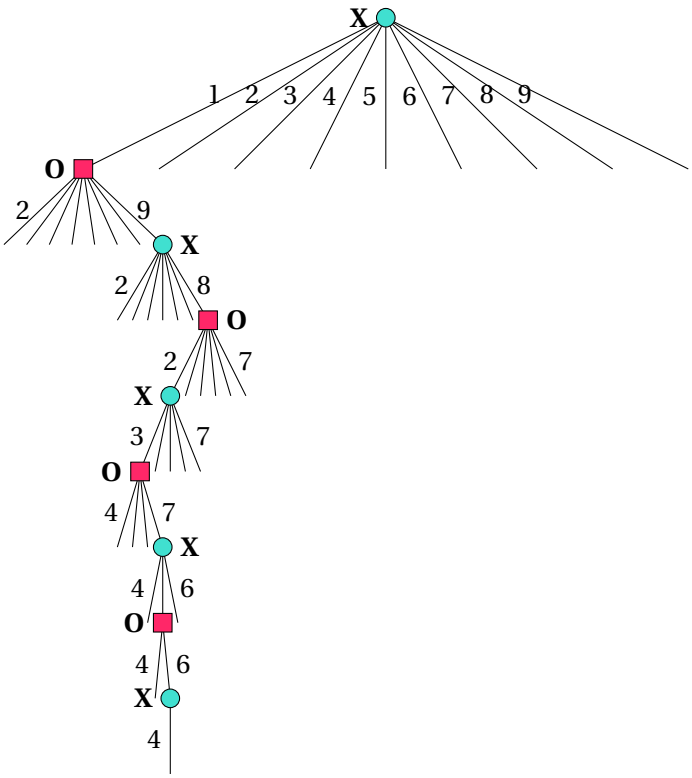
Les jeux de cartes comme le bridge ou le poker sont des jeux à information imparfaite la distribution est inconnue (aléatoire et personne n'en a connaissance puisque les cartes sont retournées lors de la distribution) et incomplète car on ne connaît pas la main des adversaires lors du jeu.

Les aventuriers du rail est un exemple de jeu de plateau à information incomplète car on ne connaît pas les objectifs des autres joueurs mais parfaite car toutes les actions passées sont connues.

■ **Définition 8 — Arbre de jeu ou forme extensive.** La représentation d'un jeu séquentiel sous la forme d'un arbre est appelée forme extensive ou arbre de jeu. Les nœuds représentent les positions du jeu. Les nœuds d'un même niveau sont contrôles par un même joueur.

Un exemple d'arbre de jeu pour une partie de morpion est donné sur la figure 1.

La représentation arborescente est fondamentale pour la plupart des raisonnements sur les jeux et en général pour l'exploration d'un ensemble de possibilités.



X	O	X
X	X	O
O	X	O

FIGURE 1 – Arbre de jeu d'une partie de morpion, partie nulle. On considère que les cases dont numérotées de 1 à 9 en ligne et en partant du haut. La position finale est donnée sous l'arbre. Le joueur à la croix *X* joue en premier car il contrôle la racine de l'arbre.

B Jeux d'accessibilité, l'exemple des jeux de Nim

■ **Définition 9 — Jeu d'accessibilité.** Un jeu d'accessibilité est un jeu à deux joueurs, à information complète et parfaite, séquentiel et pour lequel il n'y a pas de hasard. Ces jeux peuvent être modélisés par un graphe orienté biparti.

Vocabulary 1 — Impartial Game ⇜⇝ Jeu d'accessibilité. Un jeu d'accessbilité peut être qualifié d'impartial car les coups possibles ne dépendent que de la position dans le jeu et pas du joueur.

■ **Définition 10 — Jeu de Nim.** Un jeu de Nim est un jeu d'accessibilité dont il existe de nombreuses variantes. Il s'agit de déplacer, de poser ou de retirer un certain nombre d'objets simples (pièces, allumettes, graines, des billes…). Le dernier à jouer gagne ou perd (variante misère). Le jeu de Nim fait donc nécessairement un perdant et un gagnant.

Vocabulary 2 — Soustraction game ⇜⇝ Jeu de Nim ou jeu de la soustraction.

■ **Exemple 3 — Variantes du jeu de Nim.** Parmi les variantes les plus célèbres, on peut citer :

- le jeu de Marienbad (avec des cartes ou des allumettes) [**itemproductions_nim_2010**],
- le jeu des bâtonnets [a] [**fort-boyardfr_batonnets_2011**],
- le jeu de Grundy.

La figure 2 donne un exemple de jeu de Marienbad tel qu'il est présenté dans le film d'Alain Resnais. Chaque joueur peut retirer autant d'allumettes qu'il le veut sur une ligne seulement. Le perdant est celui qui retire la dernière allumette.

Ce jeu est modélisable par un graphe orienté comme l'indique la figure 3. Sur cette figure, on considère que les joueurs jouent alternativement en se déplaçant sur le graphe : un des joueurs est initialement sur la position start, quatre allumettes sont reparties sur deux rangées.

Une modélisation plus exacte peut se faire en utilisant un graphe biparti comme le montre la figure 4. Ces deux figures illustrent la même position de départ. Le graphe biparti distingue en couleur les sommets des joueurs. Sur ce graphe, on peut jouer tous les cas : le joueur rouge est le premier ou le joueur cyan est premier.

a. type Fort Boyard

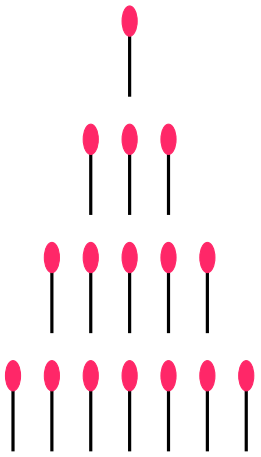


FIGURE 2 – Jeu de Marienbad avec des allumettes

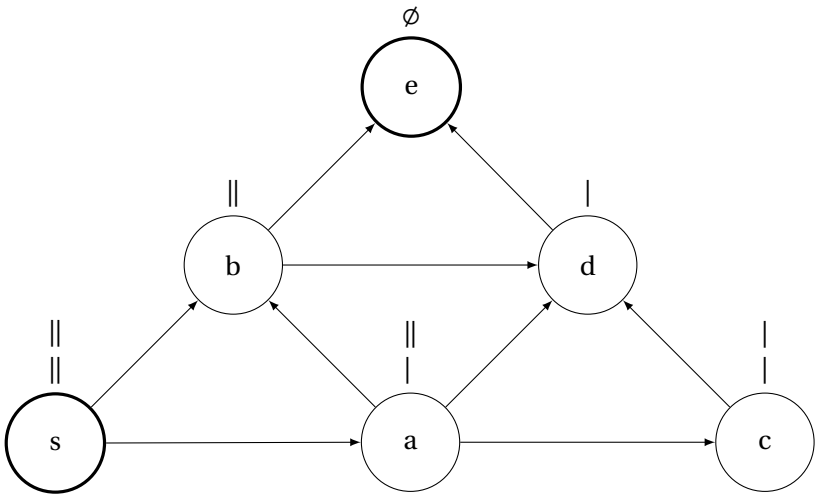


FIGURE 3 – Modélisation par graphe orienté d'une partie de jeu de Nim avec deux rangées de deux allumettes au départ. On peut jouer dessus avec un pion placé en *s* au départ. Puis chaque joueur fait avancer le pion d'un saut sur le graphe en sélectionnant un successeur en suivant les arcs. Le joueur qui se trouve en position *e* a gagné ou perdu dans la variante misère.

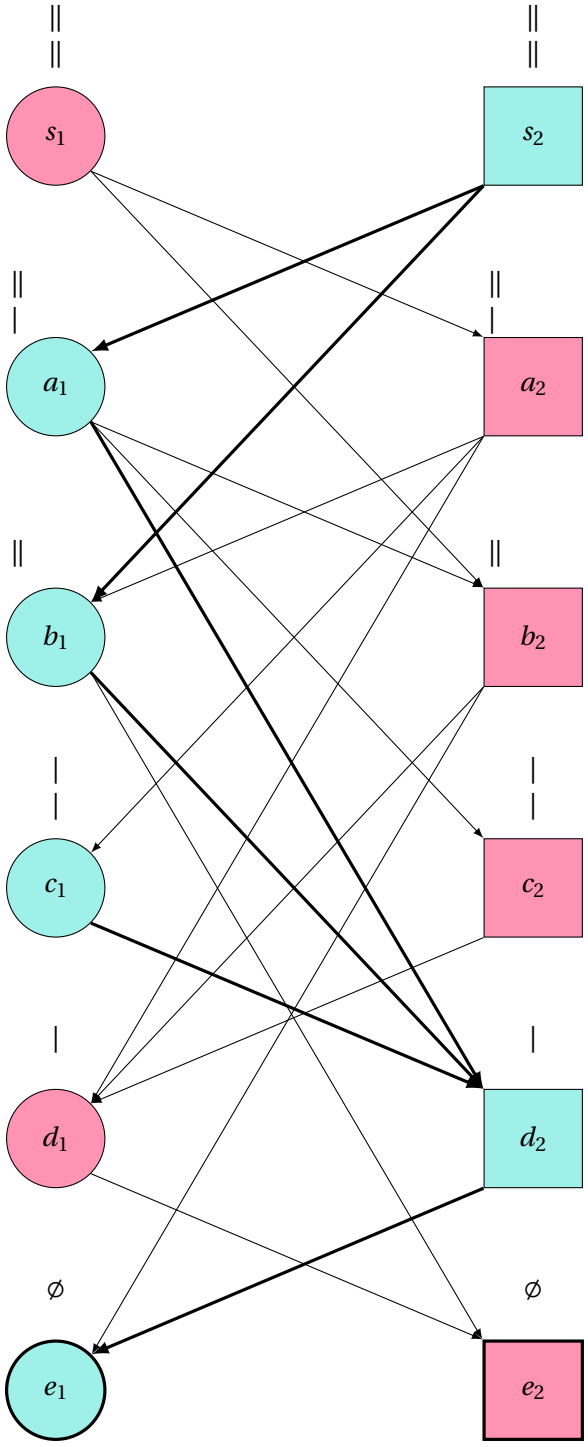


FIGURE 4 – Modélisation par graphe orienté biparti d'un jeu de Nim. Les sommets des joueurs 1 et 2 sont distingués par des cercles (1) et des carrés (2). La couleur cyan représente l'attracteur du joueur $J_a : \mathcal{A}_a = \{s_2, a_1, b_1, c_1, d_2, e_1\}$. On voit donc que a n'a pas intérêt à commencer à jouer dans cette configuration. Il en est de même pour le joueur b dont l'attracteur en rouge. C'est normal car la somme de Nim de la configuration initiale est nulle.

C Modélisation d'un jeu d'accessibilité

■ **Définition 11 — Arène de jeu.** Le graphe $G = (V_1, V_2, E)$ est nommé arène de jeu si est biparti si $G = (V = V_1 \cup V_2, E)$ est un graphe orienté biparti et $V_1 \cap V_2 = \emptyset$. Sur cette arène, le joueurs se répartissent dont les sommets : le joueur J_1 contrôle V_1, le joueur J_2 V_2.

■ **Définition 12 — Partie.** Une partie est un chemin sur l'arène de jeu : à chaque tour, le joueur J_1 en $v_i \in V_1$ choisit une arête de E dont le premier sommet est v_i et le second un sommet $v_j \in V_2$. J_2 choisit ensuite à partir de v_j le sommet suivant dans V_1. Une partie en n coups s'écrit donc $(v_0, \ldots, v_i, \ldots, v_n)$.

■ **Définition 13 — Condition de gain.** Une condition de gain pour un joueur J_i sur une arène de jeu $G = (V, E)$ est un sous-ensemble C_i^g de V_i. La partie est remportée par le joueur J_i si celui-ci visite un sommet de C_i^g en premier.

■ **Définition 14 — Condition de victoire.** Une condition de victoire d'un joueur J_i est un sous-ensemble de toutes les parties possibles $\mathcal{P}$ remportées par ce joueur. On la note :

$$C_i^v = \{\mathcal{P}, \mathcal{P} \text{ visite un sommet de } C_i^g\} \tag{1}$$

■ **Exemple 4 — Condition de gain et de victoire pour le jeu de Nim.** Pour le jeu de Nim de la figure 4, $C_1^g = \{c_1\}$ est une condition de gain pour J_1. De plus, $C_1^v = \{(s_1, b_2, c_1), (s_1, a_2, c_1)\}$ est la condition de victoire de J_1.

D Stratégies et positions

■ **Définition 15 — Stratégie sans mémoire.** Soit $G = (V, E)$ une arène de jeu. On note $V_i^{>0}$ lensemble des sommets contrôlés par le joueur $i \in \{1, 2\}$ de degré sortant non nul. Une stratégie est une application $\phi : V_i^{>0} \to V$ telle que :

$$\forall v \in V_i^{>0}, (v, \phi(v)) \in E \tag{2}$$

Cette stratégie est sans mémoire car elle ne dépend que du sommet courant et pas des sommets précédents de la partie.

(R) Une stratégie permet donc de calculer le coup à jouer. Le joueur J_i suit la stratégie ϕ lors d'une partie $\mathcal{P} = (v_0, v_1, \ldots, v_n)$ si $\forall j \in [\![0, n]\!], v_j \in V_i^{>0} \implies v_{j+1} = \phi(v_j)$

■ **Définition 16 — Stratégie gagnante.** Une stratégie ϕ est gagnante pour le joueur J_i depuis le sommet $v_0 \in V_i$ si toute partie jouée depuis v_0 par J_i en suivant ϕ est gagnante pour J_i.

■ **Définition 17 — Position gagnante.** Soit $G = (V = V_1 \cup V_2, E)$ un jeu d'accessibilité à deux joueurs. Un sommet $v \in V_i$ est appelé position gagnante pour le joueur J_i si celui-ci possède une stratégie gagnante depuis v.

■ **Exemple 5 — Position gagnante du jeu de Nim.** Sur le jeu de la figure 4, le sommet a_1 est une position gagnante pour J_1. Reste à trouver la stratégie ϕ... Le sommet b_1 n'est pas une position gagnante pour J_1.

E Attracteurs

Pour gagner une partie d'un jeu à deux joueurs, il semble donc logique de chercher les positions gagnantes et une stratégie associée. La notion d'attracteur a été développée pour construire l'ensemble des positions gagnantes. L'idée est de construire cet ensemble en partant de la condition de gain, en remontant les arc du graphe à l'envers et en ne conservant que les positions gagnantes.

■ **Définition 18 — Suite des ensembles attracteurs.** Soit $G = (V = V_1 \cup V_2, E)$ une arène d'un jeu d'accessibilité. On définit par induction la suite des attracteurs $(\mathcal{A}^1_j)_{j\in\mathbb{N}}$ du joueur J_1, c'est à dire des ensembles des sommets de V à partir desquels le joueur J_1 peut forcer la partie à arriver en C^g_1, de la manière suivante :

$$\mathcal{A}^1_0 = C^g_1 \quad \text{si } j = 0 \tag{3}$$

$$\mathcal{A}^1_{j+1} = \mathcal{A}^1_j \cup \{v \in V_1, \exists v' \in \mathcal{A}^1_j, (v,v') \in E\} \cup \{v \in V_2, \forall v' \in V, (v,v') \in E \Rightarrow v' \in \mathcal{A}^1_j\} \forall j \geqslant 0 \tag{4}$$

$$\tag{5}$$

Formulé simplement, le premier terme de cette suite est la condition de gain du joueur, c'est à dire les sommets qui lui donnent la victoire. Puis, le terme $j+1$ de la suite est l'union :

- du terme $\mathcal{A}^1_j$,
- des sommets de V_1 qu'un arc peut mener à une position gagnante de $\mathcal{A}^1_j$ de V_2,
- des sommets de V_2 qui font obligatoirement aboutir à une position gagnante de V_1.

■ **Définition 19 — Attracteur du joueur J_i.** L'attracteur du joueur i est l'ensemble des sommets d'une arène de jeu défini par :

$$\mathcal{A}^i = \bigcup_0^{+\infty} \mathcal{A}^i_j. \tag{6}$$

Théorème 1 — L'attracteur du joueur J_i contient exactement toutes les positions gagnantes de J_i.

Démonstration. On procède par récurrence sur le rang d'un sommet de G, une fonction $r : V \rightarrow$

$\mathbb{N}$ définie comme suit :

$$\forall v \in V, r(v) = \min\{j, v \in \mathcal{A}_j^i\} \tag{7}$$

Pour un sommet n'appartenant pas à l'attracteur $\mathcal{A}$, le rang est infini. Cette définition est possible car la suite $(\mathcal{A}_j^1)_{j\in\mathbb{N}}$ est croissante au sens de l'inclusion.

L'hypothèse de récurrence est la suivante : $\mathcal{H}_j$: *Pour tout $j \in \mathbb{N}$, les sommets de rang j sont des positions gagnantes du joueur J_1.*

- Initialisation $\mathcal{H}_0$: pour $j = 0$, $\mathcal{A}_0^1 = C_1^g$, donc tous les sommets de $\mathcal{A}_0^1$ sont des positions gagnantes.
- Hérédité : on suppose que, pour un certain entier naturel j, l'ensemble $\mathcal{A}_j^1$ ne contient que des positions gagnantes de J_1 ($\mathcal{H}_j$ est vraie). Considérons maintenant un élément v de l'ensemble $\mathcal{A}_{j+1}$ de rang $j+1$. Supposons de plus [1] que v n'appartient pas à $\mathcal{A}_j^1$. Il reste alors deux possibilités :
  1. Si $v \in V_1$, alors par définition de l'ensemble, il existe un arc qui amène à une position gagnante de $\mathcal{A}_j^1$. Donc, v est une position gagnante.
  2. Si $v \in V_2$, alors par définition de l'ensemble, tous les arcs de l'arène l'amène vers une position gagnante de $\mathcal{A}_j^1$. C'est donc une position gagnante.
- Conclusion : on peut donc conclure que les ensembles $\mathcal{A}_j^1$ ne contiennent que des positions gagnantes. L'attracteur $\mathcal{A}$ ne possède donc que des positions gagnantes.

■

R On peut maintenant construire une stratégie **gagnante** : la stratégie sans mémoire ϕ qui, au fur et à mesure de la partie, fait diminuer le rang de la position courante :

$$\forall v_j \in \mathcal{A}_j^1 \cap V_1, v_{j+1} = \phi(v_j), r(v_{j+1}) < r(v_j) \tag{8}$$

est gagnante. En effet, en choisissant de diminuer le rang de la position suivante, on se rapproche de la victoire.

F Algorithme de calcul de l'attracteur

Pour trouver l'attracteur d'un joueur, il faut parcourir l'arène de jeu en inversant les arcs, c'est à dire en transposant le graphe. Ainsi, en partant de la conditions de gain C_i^g et en remontant les arcs, on parvient à trouver $\mathcal{A}$.

L'algorithme 1 détaille la procédure à suivre. Cet algorithme possède une sous-fonction récursive. Comme le graphe est fini, on est cependant sûr de la terminaison. Sa correction a été donnée par la démonstration précédente : on ne fait que faire croître les ensembles $\mathcal{A}_j$ et appliquer leur définition.

Par ailleurs, l'algorithme utilise les degrés entrants du graphe transposé, mais il est tout à fait possible de raisonner sur les degrés sortant du graphe. Il m'apparaît juste plus naturel

1. sinon c'est trivial

de remonter le graphe transposé en se demandant si on peut arriver à un sommet de V_2 autrement que par un sommet de l'attracteur, plutôt qu'en se demandant si les chemins sortant d'un sommet de V_2 mènent à des sommets n'appartenant pas à l'attracteur.

Algorithme 1 Calcul de l'attracteur du joueur J_1

Entrée : g le graphe biparti de l'arène de jeu
Entrée : V_1 l'ensemble de sommets du joueur J_1
Entrée : C_g^1 la condition de gain du joueur J_1

```
1 : Fonction ATTRACTEUR(g, V_1, C_g^1)
2 :     A ← ∅                                              ▷ l'attracteur
3 :     g^t ← le transposé du graphe g                     ▷ Pour remonter le graphe
4 :     d_in ← tableau des degrés entrants du graphe transposé   ▷ Pour compter ce qui entre
5 :     pour chaque sommet v ∈ C_g^1 répéter               ▷ On part de la condition de gain
6 :         AUGMENTER_ATTRACTEUR(v, A, g^t, d_in, V_1)
7 :     renvoyer A
8 : Fonction AUGMENTER_ATTRACTEUR(v, A, g^t, d_in, V_1)
9 :     si v ∉ A alors
10 :        A ← A ∪ {v}
11 :        pour chaque voisin u de v répéter               ▷ Pour remonter le graphe
12 :            d_in[u] ← d_in[u] − 1                      ▷ On passe par ce sommet une fois depuis A
13 :            si u ∈ V_1 ou d_in[u] = 0 alors ▷ Soit u ∈ V_1 soit tous ses arcs entrant viennent de A
14 :                AUGMENTER_ATTRACTEUR(u, A, g^t, d_in, V_1)
```

★ G Solution des jeux de Nim et impartiaux --→ HORS PROGRAMME

S'il est possible de calculer les attracteurs d'un joueur, il reste néanmoins un problème de taille : comment disposer de l'arène? En effet, sur un exemple simple comme le jeu de Marienbad décrit sur la figure 4, il est relativement facile de créer le graphe associé à une arène de jeux. Cependant, dès que les dimensions augmentent, par exemple le nombre de rangées et le nombre de bâtonnets, même sur un jeu fini, il devient difficile de construire le graphe en entier. Il faut donc envisager d'autres méthodes pour trouver des stratégies gagnantes. Le nombre de Grundy permet de calculer la stratégie à adopter sans construire le graphe en entier, en connaissant uniquement la position courante du jeu, c'est à dire les nombre de bâtonnets des n piles $(x_1, \ldots, x_n)$.

L'idée ingénieuse des mathématiciens pour résoudre les jeux d'accessibilité est la suivante :

- ramener un jeu d'accessibilité à un jeu de Nim dans une position donnée, $(x_1, \ldots, x_n)$ où les x_i sont les tailles des piles,
- décomposer ce jeu en n jeux de Nim à une seule pile $G_1, G_2, \ldots, G_n$ et définir une addition sur jeux pour faire en sorte que le jeu initial soit somme des jeux à une pile.

L'addition a été trouvé par Bouton en 1901, c'est le ou exclusif.

Théorème 2 — Bouton[bouton_nim_1901]. Un position donnée $(x_1,\dots,x_n)$ d'un jeu de Nim est une position gagnante si et seulement si $x_1 \oplus x_2 \oplus \dots \oplus x_n = 0$, où $\oplus$ est l'opérateur du ou exclusif bit à bit [a].

a. Par exemple, $5 \oplus 3 = 6$

■ **Définition 20 — Nombre de Grundy d'un jeu de Nim.** Le nombre de Grundy est la somme trouvée dans le théorème 2 en utilisant le ou exclusif sur la position du jeu : $x_1 \oplus x_2 \oplus \dots \oplus x_n = 0$.

Mais on peut également la définir récursivement :

- si la pile du jeu de Nim est en position finale, le nombre de Grundy vaut 0,
- sinon, le nombre de Grundy d'une position donnée $(x_1,\dots,x_n)$ est le plus petit entier positif ou nul qui n'apparaît pas dans la liste des nombres de Grundy des positions qui suivent immédiatement la position donnée.

Ceci s'écrit parfois :

$$\gamma = mex(x_1,\dots,x_n) \tag{9}$$

où la fonction mex est le plus petit entier positif non trouvé dans un ensemble.

Le théorème 3 décrit alors précisément la stratégie gagnante.

Théorème 3 — Sprague et Grundy . Tout jeu d'accessibilité $\mathcal{J}$ est équivalent à un jeu de Nim $\mathcal{N}$.

Pour une position de $\mathcal{J}$ donnée, il existe une position de $\mathcal{N}$ dont le nombre de Grundy est γ. Cette position est équivalente à celle d'un jeu de Nim à un seul tas comportant γ allumettes.

La stratégie gagnante est celle qui consiste à choisir la position suivante de telle manière à ce que son nombre de Grundy soit nul.

■ **Exemple 6 — Utilisation du nombre de Grundy.** Ces théorèmes permettent d'affirmer que la position de départ de la figure 4 est une position perdante, car le nombre de Grundy est nul : $10_2 \oplus 10_2 = 00_2$. Il ne reste plus qu'à vous entraîner au calcul en binaire.

H Au-delà des jeux d'accessibilité, les heuristiques

Une stratégie est connue pour les jeux d'accessibilité, mais, s'ils permettent de briller dans les salons, une fois la stratégie connue, la lassitude s'installe. De nombreux autres jeux existent, comme les échecs ou le go. Pour ces jeux, il est impensable de dessiner l'arbre de jeu, la combinatoire nous indique que le nombre de parties possibles est bien trop grand. Le nombre de Shannon est une tentative pour estimer le nombre de parties différentes [2] possibles. Il vaut 10^{123} pour les échecs ce qui est plus grand que le nombre d'atomes dans l'univers observable... S'il

2. qui ont un sens

nous faut donc renoncer à explorer ces arbres de jeux d'une manière exhaustive, rien ne nous empêche néanmoins de les explorer localement.

Si on utilise un arbre de jeu exhaustif, la partie se finit lorsque la position est une feuille à laquelle est associé un gain ou un score. Lorsqu'on explore localement un arbre de jeu, les feuilles sont parfois absentes voire encore très loin de la position. C'est pourquoi le développement de ces algorithmes s'appuie sur des heuristiques capables d'estimer le gain d'une position sans disposer de l'intégralité de l'arbre de jeu.

■ **Définition 21 — Heuristique.** Une heuristique[a] est une approche de résolution de problème à partir de connaissances incomplètes. Une heuristique doit permettre d'aboutir en un temps limité à des solutions néanmoins acceptables mêmes si elles ne sont pas nécessairement optimales.

a. Je trouve en grec ancien.

I Minimax et les heuristiques

L'algorithme Minimax associé à une heuristique permet de développer des stratégies sans avoir à explorer tout l'arbre de jeu. Il découle naturellement du théorème du même nom démontré par Von Neumann en 1926.

L'algorithme 2 détaille la procédure Minimax. Le principe est le suivant : on construit un arbre de jeu incomplet comme celui-de la figure 5. La hauteur de l'arbre est un entier fixé qui limite la profondeur de l'exploration de l'algorithme. La complexité s'en trouve ainsi améliorée.

Chaque niveau de l'arbre est contrôlé par un seul joueur. Comme, pour les jeux d'accessibilité considérés, les gains de l'un sont les pertes de l'autre, on choisit de nommer les joueurs J_{max} en rouge et J_{min} en cyan. Le but du jeu est de maximiser le gain pour J_{max}, son score est donc positif, et, symétriquement, minimiser le gain pour J_{min} dont le score est donc négatif.

■ **Définition 22 — Définition du score de joueurs pour une position donnée.** Le jeu associe à chaque feuille de l'arbre minimax un gain qui devient le score du joueur qui atteint cette feuille. Si la feuille est en position p alors on choisit de noter ce score $s(p) \in \mathbb{R}$. À partir du score associé aux feuilles, on peut définir récursivement le score maximal ou minimal associé à un nœuds internes p de l'arbre minimax comme suit :

$$\mathcal{S}(p) = \begin{cases} s(p) & \text{si } p \text{ est une feuille} \\ \max\{\mathcal{S}(f), f \text{ fils de } p\} & \text{si } p \text{ est contrôlé par } J_{max} \\ \min\{\mathcal{S}(f), f \text{ fils de } p\} & \text{si } p \text{ est contrôlé par } J_{min} \end{cases} \tag{10}$$

Ainsi, à la fin de la partie, si J_{max} en position p effectue les choix décrit ci-dessus, son score final sera au moins $\mathcal{S}(p)$. De même, le score de J_{min} en position p sera au plus $\mathcal{S}(p)$. Ceci peut se démontrer par récurrence.

Le score d'un joueur ne peut se calculer tel que décrit ci-dessus puisqu'on ne connaît pas l'intégralité de l'arbre de jeu. L'algorithme 2 suppose donc qu'on connaît une heuristique $\mathcal{H}$ pour estimer le score d'une position d'un nœud intermédiaire. Cette heuristique est une fonc-

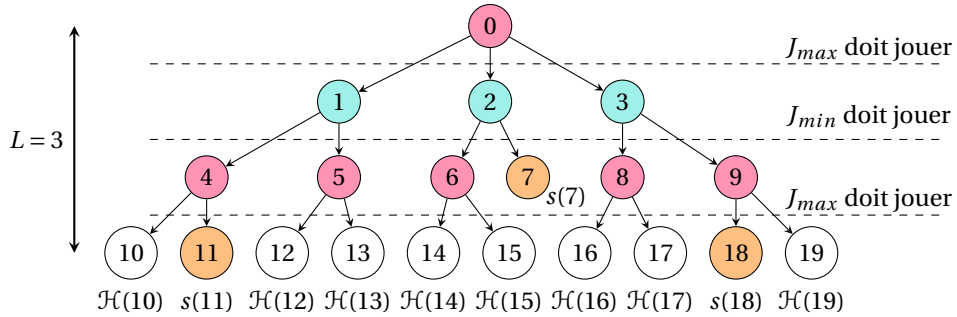


FIGURE 5 – Exemples d'arbre minimax. Chaque niveau est contrôlé par un seul joueur, J_{max} en rouge et J_{min} en cyan. La hauteur de l'arbre L est telle que seule une partie de l'arbre de jeu est accessible. Certaines feuilles sont visibles (en orange, c'est l'automne). L'arbre minimax s'achève donc parfois sur des nœuds internes pour lesquels on donne une estimation du gain (non coloré).

tion de la position p dans l'arbre et sa valeur est un nombre réel $\mathcal{H}(p)$.

■ **Exemple 7 — Calcul des scores sur un arbre Minimax.** La figure 6 superpose les scores calculés par l'algorithme Minimax sur chaque nœud. Le joueur J_{max} peut donc espérer au plus un score de 6 s'il se trouve en position 0.

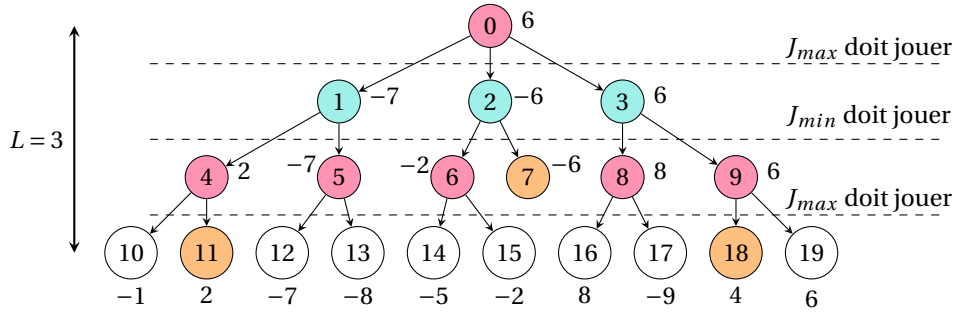


FIGURE 6 – Exemples d'arbre minimax complété avec les scores. Chaque niveau est contrôlé par un seul joueur, J_{max} en rouge et J_{min} en cyan.

Algorithme 2 Minimax

Entrée : p un position dans l'arbre de jeu (un nœu de l'arbre)
Entrée : s la fonction de score sur les feuilles
Entrée : $\mathcal{H}$ l'heuristique de calcul du score pour un nœud interne
Entrée : L la profondeur maximale de l'arbre Minimax

```
1 : Fonction MINIMAX(p, s, H , L)
2 :     si p est une feuille alors
3 :         renvoyer s(p)
4 :     si L = 0 alors
5 :         renvoyer H(p)                          ▷ On arrête d'explorer, on estime
6 :     si p est contrôlé par J_max alors
7 :         M ← −∞
8 :         p_M un nœud vide
9 :         pour chaque fils f de p répéter
10 :            v ←MINIMAX(f,s,H, L − 1)            ▷ v est un score de J_min
11 :            si v > M alors
12 :                M ← v
13 :                p_M = f
14 :        renvoyer M, p_M        ▷ Valeur maximale trouvée et la racine de cette solution
15 :    sinon
16 :        m ← +∞
17 :        p_m un nœud vide
18 :        pour chaque fils f de p répéter
19 :            v ←MINIMAX(f,s,H, L − 1)            ▷ v est un score de J_max
20 :            si v < m alors
21 :                m ← v
22 :                p_m = f
23 :        renvoyer m, p_m        ▷ Valeur minimale trouvée et la racine de cette solution
```

J Élagage $\alpha\beta$ sur un arbre Minimax --→ HORS PROGRAMME

Selon les situations considérées, limiter la profondeur d'exploration de l'arbre de jeux peut s'avérer être insuffisant pour réduire la complexité du problème. L'élagage $\alpha\beta$ est une technique pour ne pas explorer certaines branches de l'arbre qui n'ont pas besoin de l'être. Le principe est détaillé sur la figure 7.

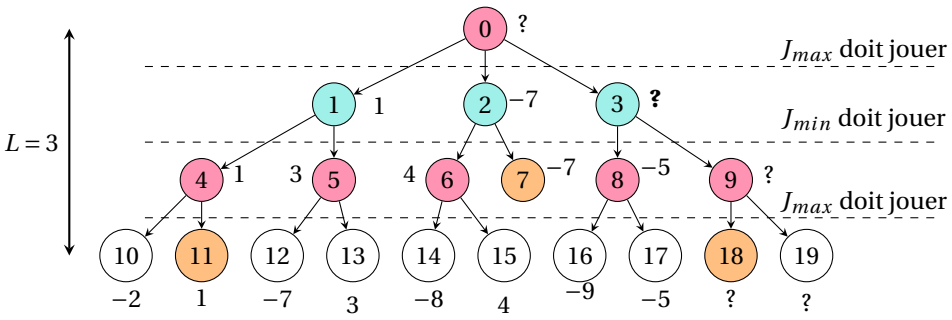


FIGURE 7 – L'élagage $\alpha\beta$ permet sur cet exemple d'éviter l'exploration complète du sous-arbre du nœud numéro 3. En effet, comme c'est un nœud de J_{min}, que le premier nœud du niveau a un score de 1 et que le score calculé du nœud 8 vaut -5, alors on comprend que J_{min} remontera au moins -5 et que J_{max} ne choisira pas ce nœud numéro 3 car ce n'est pas le maximum du niveau.

On distingue deux types d'élagage possible, les types α et β comme le montre les figures 8 et 9. Pour améliorer la complexité de l'algorithme Minimax, on peut choisir d'élaguer comme le montre l'algorithme 3.

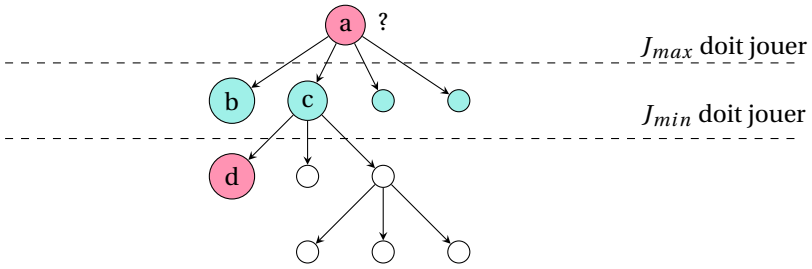


FIGURE 8 – L'élagage α : supposons que $\mathcal{S}(b)$ ait déjà été calculé par l'algorithme Minimax. Si $\mathcal{S}(b) \geqslant \mathcal{S}(d)$, alors il est inutile d'explorer le sous-arbre c : J_{max} ne choisira pas le nœud c, il lui préfèrera b.

Algorithme 3 Minimax avec élagage $\alpha\beta$

Entrée : p un position dans l'arbre de jeu (un nœu de l'arbre)
Entrée : s la fonction de score sur les feuilles
Entrée : $\mathcal{H}$ l'heuristique de calcul du score pour un nœud interne
Entrée : L la profondeur maximale de l'arbre Minimax
Entrée : α le niveau de coupure α ▷ $-\infty$ au démarrage
Entrée : β le niveau de coupure β ▷ $+\infty$ au démarrage

```
1 : Fonction MINIMAX_αβ(p, s, H , L, α, β)
2 :     si p est une feuille alors
3 :         renvoyer s(p)
4 :     si L = 0 alors
5 :         renvoyer H(p)                          ▷ On arrête d'explorer, on estime
6 :     si p est contrôlé par J_max alors
7 :         M ← −∞
8 :         p_M un nœud vide
9 :         pour chaque fils f de p répéter
10 :            v ←MINIMAX_αβ(f,s,H, L − 1, α, β)          ▷ v est un score de J_min
11 :            si v > M alors
12 :                M ← v
13 :                p_M = f
14 :            si v ⩾ β alors
15 :                renvoyer M                     ▷ Élagage de type β
16 :            α ← max(α, M)                      ▷ Mise à jour du niveau de l'élagage
17 :        renvoyer M, p_M          ▷ Valeur maximale trouvée et la racine de cette solution
18 :    sinon
19 :        m ← +∞
20 :        p_m un nœud vide
21 :        pour chaque fils f de p répéter
22 :            v ←MINIMAX_αβ(f,s,H, L − 1, α, β)          ▷ v est un score de J_max
23 :            si v < m alors
24 :                m ← v
25 :                p_m = f
26 :            si v ⩽ α alors
27 :                renvoyer m                     ▷ Élagage de type α
28 :            β ← min(β, m)                      ▷ Mise à jour du niveau de l'élagage
29 :        renvoyer m, p_m          ▷ Valeur minimale trouvée et la racine de cette solution
```

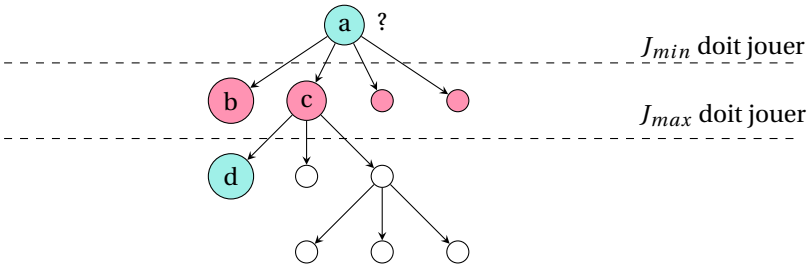


FIGURE 9 – L'élagage β : supposons que $\mathcal{S}(b)$ ait déjà été calculé par l'algorithme Minimax. Si $\mathcal{S}(b) \leqslant \mathcal{S}(d)$, alors il est inutile d'explorer le sous-arbre c : J_{min} ne choisira pas le nœud c, il lui préfèrera b.

K A* pour trouver un chemin

Si le jeu que l'on considère ne peut pas être modélisé selon un arbre Minimax, alors la situation n'est pas désespérée car il nous reste encore les graphes. Il est toujours possible de modéliser l'évolution d'un jeu comme une succession d'état, chaque coup joué permettant de passer d'un état à un autre. Développer une stratégie dans un jeu modélisé par un graphe à état revient donc à trouver un chemin d'un sommet à un autre dans un graphe. Or, nous sommes déjà bien outillés pour faire cela.

L'algorithme A*[3] est un algorithme couteau suisse qui peut être considéré comme un algorithme de Dijkstra muni d'une heuristique : là où Dijkstra ne tient compte que du coût du chemin déjà parcouru, A* considère ce coût et une heuristique qui l'informe sur le reste du chemin à parcourir. Il faut bien remarquer que le chemin qu'il reste à parcourir n'est pas nécessairement déjà exploré : il est rarement possible d'explorer tout le graphe à état d'un jeu. Si l'heuristique pour évaluer le reste du chemin à parcourir est bien choisie, alors A* converge aussi vite voire plus vite que Dijkstra[**unswmechatronics_dijkstras_2013**].

■ **Définition 23 — Heuristique admissible.** Une heuristique $\mathcal{H}$ est admissible si pour tout sommet du graphe d'état, $\mathcal{H}(s)$ est une borne inférieure de la plus courte distance séparant le sommet de départ du sommet d'arrivée.

■ **Définition 24 — Heuristique cohérente.** Une heuristique $\mathcal{H}$ est cohérente si pour tout arc (s,p) du graphe d'état $G = (V,E,w)$, $\mathcal{H}(s) \leqslant \mathcal{H}(p) + w(s,p)$.

■ **Définition 25 — Heuristique monotone.** Une heuristique $\mathcal{H}$ est monotone si l'estimation du coût **total** du chemin ne décroît pas lors du passage d'un sommet à ses successeurs. Pour un chemin $(s_0, s_1, \ldots, s_n)$, on $\forall 0 \leqslant i < j \leqslant n, c(s_j) \geqslant c(s_i)$.

Soit $G = (V,E,w)$ un graphe orienté pondéré. Soit d la fonction de distance utilisée par

3. prononcer *A étoile* ou *A star*

l'algorithme de Dijkstra (cf. algorithme **??**). A*, muni d'une fonction h permettant d'évaluer l'heuristique, calcule alors le coût total pour aller jusqu'à un sommet p comme suit :

$$c(p) = d(p) + h(p) \tag{11}$$

Le coût obtenu n'est pas nécessairement optimal, il dépend de l'heuristique.

Supposons que l'on cherche le chemin le plus court entre les sommets s_0 et p. Supposons que l'on connaisse un chemin optimal entre s_0 et un sommet s. Alors on peut écrire que le coût total vers le sommet p vaut :

$$\begin{aligned} c(p) =& d(p) + h(p) && (12)\\ =& d(s) + w(s,p) + h(p) && (13)\\ =& d(s) + h(s) + w(s,p) - h(s) + h(p) && (14)\\ =& c(s) + w(s,p) - h(s) + h(p) && (15) \end{aligned}$$

Ainsi, on peut voir l'algorithme A* comme un algorithme de Dijkstra muni :

- de la distance $\tilde{d} = c$,
- et de la pondération $\tilde{w}(s,p) = w(s,p) - h(s) + h(p)$.

L'algorithme 4 donne le détail de la procédure à suivre.

Algorithme 4 A*

Fonction ASTAR($G = (V, E, w)$, a) ▷ Sommet de départ a
  $\Delta \leftarrow a$
  $\Pi \leftarrow$
  $\tilde{d} \leftarrow$ l'ensemble des distances au sommet a
  $\forall s \in V, \tilde{d}[s] \leftarrow \tilde{w}(a,s)$ ▷ Le graphe est partiel, l'heuristique fait le reste
  tant que $\bar{\Delta}$ n'est pas vide **répéter** ▷ $\bar{\Delta}$: sommets dont la distance n'est pas connue
    Choisir u dans $\bar{\Delta}$ tel que $\tilde{d}[u] = \min\left(\tilde{d}[v], v \in \bar{\Delta}\right)$
    $\Delta = \Delta \cup \{u\}$
    pour $x \in \bar{\Delta}$ **répéter** ▷ Ou bien $x \in \mathcal{N}_G(u) \cap \bar{\Delta}$, pour tous les voisins de u dans $\bar{\Delta}$
      si $\tilde{d}[x] > \tilde{d}[u] + \tilde{w}(u,x)$ **alors**
        $\tilde{d}[x] \leftarrow \tilde{d}[u] + \tilde{w}(u,x)$
        $\Pi[x] \leftarrow u$ ▷ Pour garder la tracer du chemin le plus court
  renvoyer $\tilde{d}$, Π