IA : Intelligence artificielle

Comment gagner au morpion à tous les coups grâce à l'intelligence artificielle ?

Un Apprentissage Par Problème (APP) destiné aux étudiants du module Algorithmique avancée 2

Février 2025

Présentation du problème

Le but de ce projet de groupe est de mettre en place un algorithme pour développer un jeu de morpion doté d'une capacité à prendre des décisions. Nous allons introduire une intelligence artificielle (IA) pour pouvoir jouer contre un ordinateur.

Lorsque nous regardons le déroulement d'une partie, cela peut être vu comme l'élaboration d'un arbre de possibilités.

Lorsque l'ordinateur doit choisir son prochain coup, il parcourt toutes les différentes possibilités et choisira la meilleure. Mais comment l'ordinateur fait-il pour décider ? L'ordinateur parcourt toutes les possibilités et choisit la meilleure solution. Cette étape de prise de décision fait appel à de « l'intelligence artificielle : IA ».

Lorsque l'IA doit décider son prochain coup, si nous limitions la recherche à une profondeur de 1, cela veut dire que l'ordinateur va se restreindre à regarder le prochain coup uniquement.
L'image suivante montre comment l'ordinateur va choisir ce prochain coup. En haut se trouve l'état courant du jeu (on l'appelle aussi configuration courante).

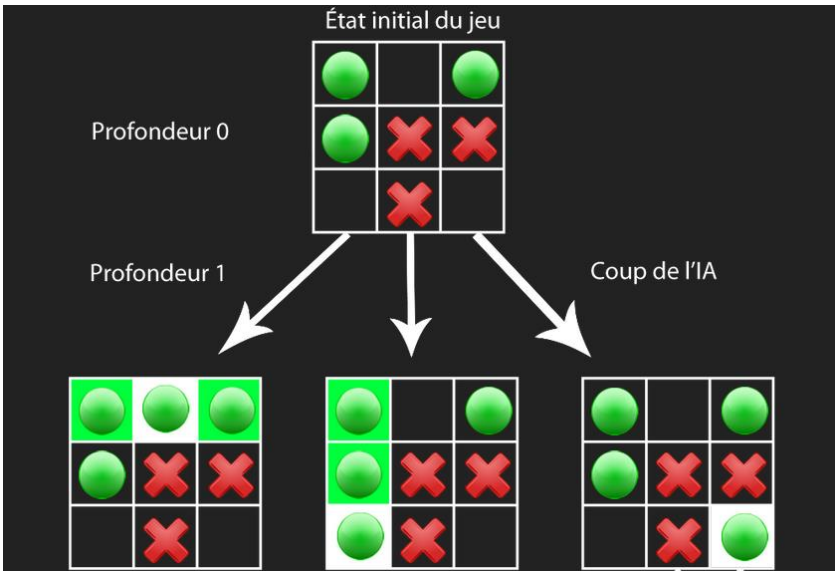

Figure 1- Arbre de décision qui illustre toutes les possibilités à partir d'une situation initiale

Pour le jeu qui nous concerne, on part d'une *situation courante* et les règles du jeu déterminent la liste des nouvelles situations de jeu atteignables à partir de celle-ci.

Chacune de ces situations, si elle est choisie par le joueur, devient la prochaine *situation courante* à partir de laquelle l'autre joueur devra à son tour choisir la prochaine situation. Ensuite, la nouvelle situation devient la *situation courante* du prochain tour de jeu et ainsi de suite jusqu'à la fin de la partie.

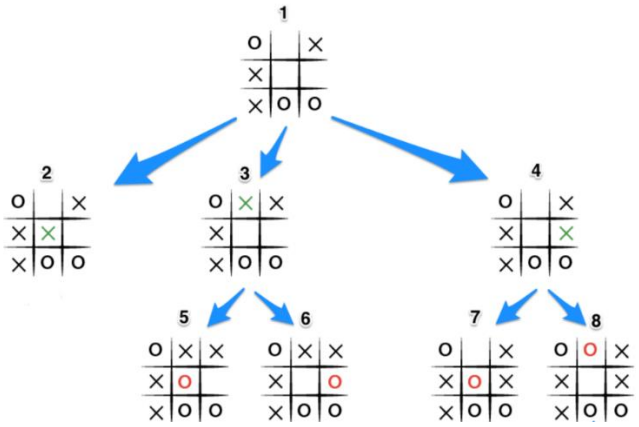

Figure 2 - Ebauche d'un arbre de jeu

Chaque *situation courante* propose donc un ensemble d'embranchements que peut choisir le joueur, et cette situation se reproduit à chaque tour de jeu. On peut donc représenter l'ensemble de toutes les parties possibles à partir d'une *situation courante* initiale sous la forme d'un graphique représentant toutes les situations possibles.

Principe de l'algorithme

Puisque le jeu est *à information complète* chaque joueur a la possibilité (théorique) de calculer ces possibilités. Le problème qui se pose au joueur est donc de choisir un chemin qui l'amène vers la meilleure situation finale possible, c'est-à-dire, si possible vers une victoire. La question qui se pose est donc :

Quel coup jouer maintenant pour gagner plus tard ?

L'objectif est donc la qualité de la situation visée "*dans le futur*" et non pas celle atteinte immédiatement, et cette qualité doit être la meilleure possible. Comment faire ? Comme tout joueur humain le fait, l'algorithme va "*calculer des coups à l'avance*" et prendre sa décision en fonction de ces calculs.

1. Le premier principe de l'algorithme que vous allez programmer est donc de déployer l'ensemble des situations de jeu et d'attribuer une note aux situations atteintes à la fin de l'arbre. Cette note doit refléter la qualité de la situation. Elle doit donc être croissante avec la qualité de la situation : plus la situation est favorable, plus la note est haute. On appelle **fonction d'évaluation**, la fonction d'attribution de la note à une situation. La note attribuée correspond au gain espéré si l'on atteint cette situation.

2. Le second principe de votre algorithme est de propager vers le haut les valeurs qui ont été attribuées aux situations qui sont en fin de graphique. Le but est d'attribuer, par propagation, une note aux situations présentes au sein de l'arbre de jeu, jusqu'à la situation initiale située tout en haut de l'arbre.

Illustrons ceci à travers un exemple. A partir d'une situation, on peut représenter toutes les évolutions possibles du jeu. On arrête lorsque la partie s'arrête, c'est-à-dire soit lorsque le joueur a gagné, soit lorsque l'IA a gagné ou lorsqu'il y a égalité.
Une fois que nous avons dressé l'arbre du jeu, on peut attribuer différentes notes à l'état final du jeu suivant le résultat :

- 1000 si l'IA a gagné
- 0 s'il y a égalité
- -1000 si le joueur a gagné

Remarque : ces valeurs sont arbitraires. On aurait pu mettre n'importe quelle autre valeur, le but étant de donner une valeur forte à une victoire de l'IA, une valeur faible pour une défaite et une valeur intermédiaire pour un résultat nul.

Décision pour le cas d'une configuration à deux coups (profondeur 2)

L'image suivante montre les possibilités avec deux coups successifs. À présent, l'ordinateur (qui joue avec les ronds) aura la connaissance des deux prochains coups possibles pour faire son choix. Déjà, nous remarquons que si l'IA choisit le coup de droite (en partant de tout en haut), nous pourrons gagner la partie en jouant ensuite le coup de gauche (on aura trois X).

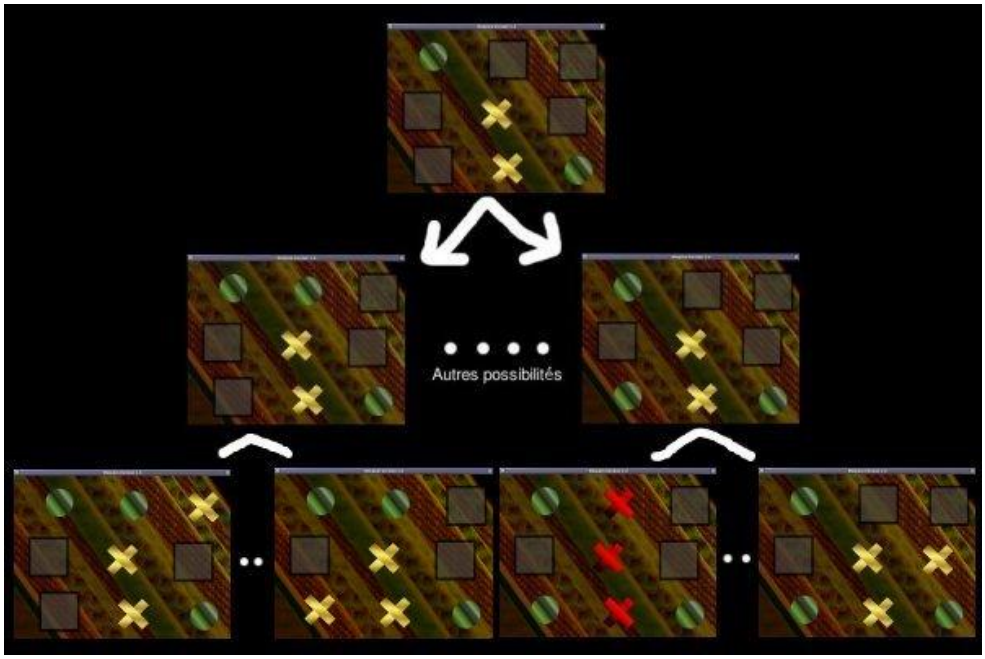

Figure 3 - Décision pour une profondeur de 2

Donc si nous utilisons le calcul de l'évaluation de la situation, la dernière image deviendra :

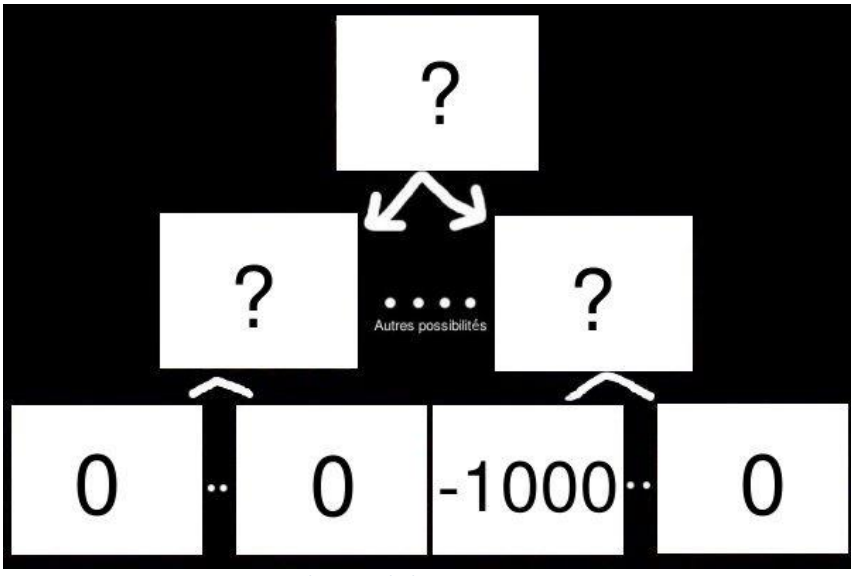

Figure 4 - Evaluation de la situation au premier coup

La fonction d'évaluation nous donne une valeur pour chaque configuration du jeu. Mais quelle valeur faut-il donner aux évaluations de l'étage du milieu ?

La solution est extrêmement logique. Ces évaluations représentent les choix que ferait l'adversaire de l'ordinateur (généralement un humain, mais cela pourrait être un autre ordinateur). Bien sûr, l'IA va supposer que l'adversaire va choisir le meilleur coup possible. Donc pour chaque nœud qui représente le choix de l'adversaire, le coup qu'il choisirait devrait mettre l'IA dans la plus grande difficulté (donc devrait avoir le score le plus petit).

Donc en supposant cela, il est logique que la valeur des nœuds de l'étage du milieu soit égale au **minimum** des valeurs de leurs fils. On obtient donc :

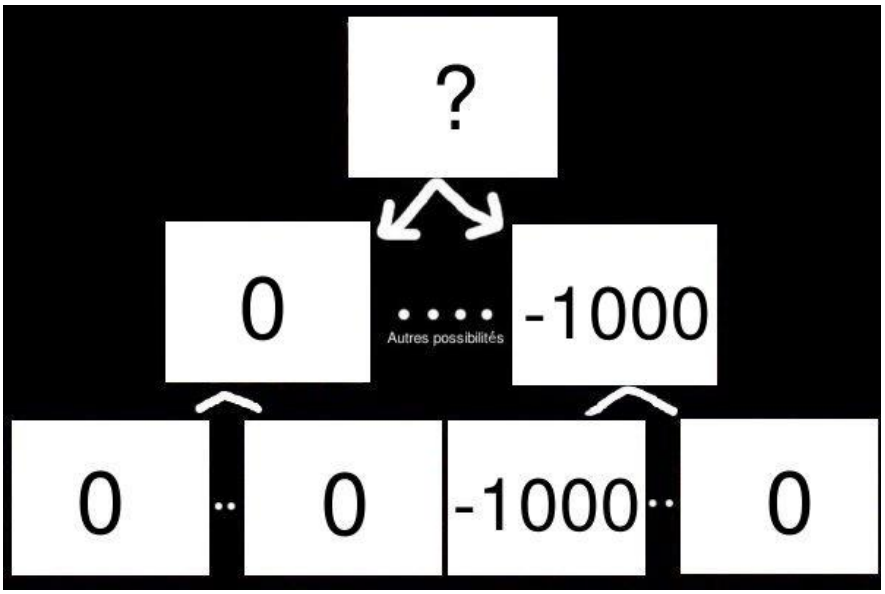

Figure 5 - Evaluation des valeurs au premier coup

Donc pour terminer, l'IA aura comme choix le coup de gauche qui mènera vers une évaluation de 0 ou le coup de droite qui mène vers une évaluation de -1000. Si vous avez bien suivi, l'IA va bien sûr devoir choisir le coup de gauche. La valeur de ce nœud est donc le **maximum** de ses fils. Voici ce que cela donne :

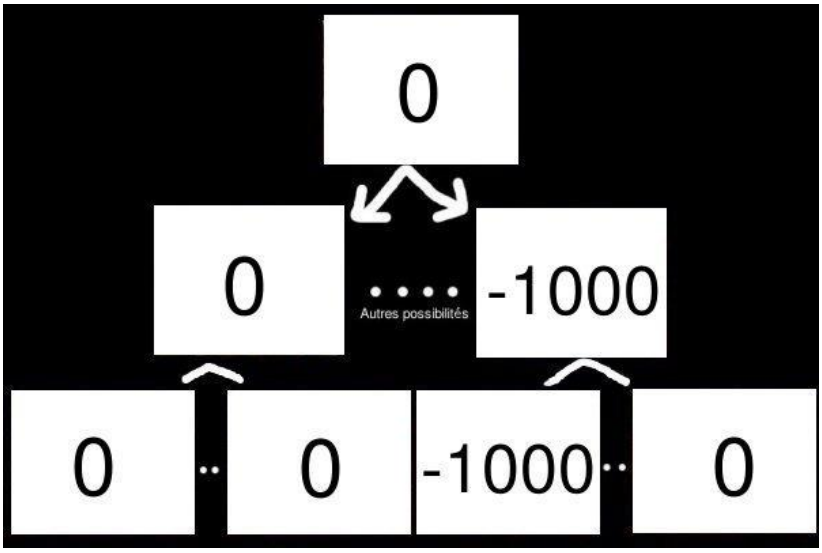

Figure 6 - Evaluation de la configuration initiale

Finalement, l'idée est de parcourir un certain nombre de configurations à partir d'une configuration initiale. Lorsqu'on arrive au bout de l'arbre, on utilise la fonction d'évaluation pour donner une valeur à la configuration.
Ensuite, pour les nœuds qui ne sont pas au bout de l'arbre, on utilise soit le minimum des valeurs des fils si le nœud représente un choix de l'adversaire (c'est donc un nœud Min), soit le maximum des valeurs des fils si le nœud représente un choix de l'IA (c'est donc un nœud Max).

Quand c'est au joueur de jouer, la note de la situation courante est donc **le max des notes des situations suivantes**.

Quand c'est à l'adversaire de jouer, la note de la situation courante est donc **le min des notes des situations suivantes**.

En fonction de l'évaluation qu'il donne au prochain coup, l'ordinateur devra donc jouer le coup qui a le score le plus élevé. C'est ainsi qu'on utilise notre intelligence artificielle.

Vous devez donc proposer un algorithme qui, en tenant compte de l'intérêt opposé de l'adversaire, attribue la meilleure note possible à toutes les situations de l'arbre de jeu. Grâce à ça vous devrez fournir un programme qui simule une partie de Morpion avec une intelligence artificielle.

L'outil que vous avez à produire devra à chaque tour permettre de :

- Simuler une partie de Morpion contre l'ordinateur
- Afficher l'ensemble des scores d'évaluation d'une génération donnée (ainsi que leur minimum et maximum)
- Afficher une suite idéale de coups pour que l'ordinateur gagne (depuis la situation courante jusqu'à la fin de l'arbre)
- Afficher une suite idéale de coups pour que l'adversaire gagne (depuis la situation courante jusqu'à la fin de l'arbre)
- Augmenter/baisser le niveau difficulté en fonction de la profondeur jusqu'à laquelle l'ordinateur devra calculer les coups à l'avance (plus la profondeur est élevée plus l'ordinateur mettra du temps !)
- Simuler une partie joueur contre joueur (sans IA) ou ordinateur contre ordinateur
- [Bonus] Essayer de généraliser votre programme à un jeu où il faut aligner cette fois quatre items identiques (au lieu de 3) pour gagner, comme le célèbre puissance 4. Qu'est-ce que ça change au niveau du calcul des possibilités ? de leur représentation ? De la fonction d'évaluation ?

Une fonction d'évaluation exploitable pour le morpion est donnée en annexe mais n'hésitez pas à faire votre propre fonction…

Les ressources pour traiter la situation-problème

MIMOS: Algorithmique S1-S2 (INTEGRAL)

Module 29 – Les arbres généraux : https://courses.ionisx.com/courses/ref/m202/x/course/

Module 31 – Les arbres généraux parcours :
https://courses.ionisx.com/courses/ref/m204/x/course/

Liens

L'algorithme présenté dans l'énoncé du problème est l'algorithme MinMax :
https://fr.wikipedia.org/wiki/Algorithme_minimax

Jouer au morpion en ligne, pour les personnes qui ont oublié ce que c'était !
https://papergames.io/fr/morpion

Informations complémentaires

Il y a un aspect de l'algorithme qu'il ne faut pas oublier : **la complexité**. En effet, la fonction d'évaluation peut être tellement compliquée que le simple nombre d'appels suffirait à rendre votre algorithme inutilisable. Prenons le simple jeu du morpion, si on calcule la taille de l'arbre, on obtient 9! nœuds (ce qui est égal à 362 880).

Il n'y a pas de bonne réponse à la profondeur du parcours qu'il faudrait utiliser. Généralement, il faut utiliser la plus grande profondeur possible mais attendre pendant 1h que l'ordinateur calcule son coup, c'est rédhibitoire pour n'importe qui !

C'est d'ailleurs l'une des raisons pour lesquelles on vous demande de définir trois niveaux de difficulté de l'IA. En fait, ce sera la profondeur (c'est-à-dire le nombre de coups à l'avance qu'on voudra anticiper) qui va être la définition de « l'intelligence ». Essayez plusieurs valeurs et trouvez le bon équilibre.

La fonction d'évaluation est fondamentale. Si cette fonction est mal établie, alors l'IA ne va pas jouer correctement. Pour beaucoup de jeux, cette fonction sera très difficile à mettre en place. Heureusement, pour le morpion, il existe beaucoup de ressources en ligne qui peuvent vous aider à trouver une fonction d'évaluation plus subtile que la fonction qu'on utilise dans l'énoncé et qui ne fournit que trois valeurs : -1000, 0 et 1000.

Quand on évalue une disposition de pions non finale (ça veut dire que la partie est en cours), Il est d'usage de partir de ces valeurs de base et d'ajouter un coefficient en fonction de la situation (on peut même additionner/soustraire une valeur numérique voire calculer une nouvelle valeur en fonction de la disposition des X et des O). Cela rend le programme plus subtil.

Une possibilité pour le morpion est cette évaluation pour chaque configuration :

```
Si la partie est terminée alors
    Si c'est une égalité, alors on retourne 0
    Sinon
        Si l'ordinateur a gagné, alors retourne 1000 (le maximum possible);
        Sinon l'ordinateur a perdu, alors retourne -1000 (le minimum
possible);
        Fin Si
    Fin Si
Sinon
    Soit la somme = 0
    Pour chaque ligne, chaque colonne, chaque diagonale,
       S'il y a au moins un pion de chaque joueur, on ne fait rien
       Sinon
            S'il y a 2 pions de l'adversaire, on enlève 30 à somme
            sinon s'il y a 2 pions du joueur de l'IA, on ajoute 30 à somme
            sinon s'il y a 1 pion de l'adversaire, on enlève 10 à somme
            sinon s'il y a 1 pion de l'IA, on ajoute 10 à somme
            Fin Si
       Fin Si
    Fin Pour
Fin Si
```

Les possibilités sont nombreuses et là encore c'est à vous de trouver la fonction qui optimise votre programme.

Calendrier du traitement de l'APP :

Timing séance « aller » et travail individuel

Phases et Etapes		Tâches
Phases A séance « ALLER »	**1** 10min	**Organiser l'équipe :** ▪ Se répartir les fonctions indispensables (voir page 11) ▪ *Le barreur prend connaissance des étapes à parcourir et garde le cap* ▪ *Le gardien du temps s'engage à surveiller le timing*
	2 10min	**Prendre connaissance du document fourni :** ▪ Chacun effectue un premier survol du cahier pour se familiariser avec le contenu
	3 10min	**Comprendre et clarifier le problème :** à partir de la p. 3 : ▪ Quel est au juste le problème que nous allons traiter ? ▪ *Le scribe commence à noter ce qui apparaît dans les échanges (mots-clés, concepts, idées, …)*
	4 30min	**Etablir ensemble des pistes pour traiter le problème :** ▪ Etablir une liste de questions pertinentes auxquelles il faudra répondre ▪ Faire le point sur ce que l'équipe connaît (et ne connaît pas) ▪ Le cas échéant, établir une liste de simplifications, de restrictions en vue de limiter la portée du problème (si nécessaire, voir avec le tuteur) ▪ Etablir une liste des productions attendues ▪ Envisager différentes pistes pour avancer dans le traitement ▪ *L'activateur lance et relance la discussion quand c'est nécessaire*
	5 20min	**Préciser les acquis d'apprentissage :** ▪ Que faut-il (ré-)apprendre / découvrir pour traiter le problème ? ▪ A quelles questions chacun de nous devra-t-il être capable de répondre à la fin de la séance « RETOUR » ? ▪ Que faudra-t-il être capable de faire ?
	6 15min	**Etablir un plan d'action :** ▪ Déterminer les informations à recueillir pour confirmer ou invalider les pistes énumérées ▪ Dresser la liste des tâches à accomplir et des livrables à préparer par chacun <u>avant</u> la prochaine séance, ... ▪ *Le secrétaire note ce qui est décidé et s'arrange pour le communiquer aux autres membres de l'équipe*

Phase B Travail Indiv.	**7** de 6 à 10 h	**Travail individuel :** ▪ Mettre en œuvre le plan d'action établi à l'étape 6 : **chacun effectue le travail décidé et prépare ce qu'il va apporter à la séance « Retour »**

Phases C Séance « RETOUR »		(détails p. 9)

Calendrier du traitement de l'APP :

Timing séance « retour »

Phases et Etapes		Tâches
Phase C Séance « RETOUR »	**8** 10min	**Organiser l'équipe :** ▪ Qui fait quoi (fonctions) ? Faut-il changer les attributions ? De nouvelles fonctions doivent-elles être attribuées ? ▪ Quelle production faut-il / choisissons-nous de produire ? → livrable(s) (si nécessaire, confirmer avec le tuteur) ▪ **Planification** : que faut-il faire à l'étape suivante et comment organiser au mieux le temps disponible ? (+ consignes pour le gardien du temps)
	9 60min	**Valider les apprentissages, les solutions, les livrables** ▪ Mettre en commun ce que chacun a étudié, préparé, apporté ▪ Examiner les réponses aux questions formulées lors de la séance « aller » ▪ Proposer des réponses / solutions à la situation-problème ▪ Préparer le(s) livrable(s) ▪ Valider collectivement ▪
	25min	**QCM noté**
	25min	**Correction QCM**

A propos des fonctions

Les différentes fonctions utiles/nécessaires à un bon travail en équipe sont présentées sous la forme de fiches/cartes déposées sur la table par le tuteur. Au verso, ces cartes indiquent en quoi consiste la fonction.

Les membres de l'équipe examinent chacune de ces cartes (surtout le verso) et se répartissent les fonctions **en commençant par celles qui sont indispensables**. Il n'est pas nécessaire que les autres fonctions soient attribuées, surtout lorsque c'est un premier APP pour l'équipe. Les cartes sont disposées sur la table de manière à permettre à chacun (et au tuteur) de voir qui prend en charge quelle(s) fonction(s).

Il est néanmoins important que les participants comprennent l'utilité des différentes fonctions ; comme il s'agit d'un premier APP, il n'est pas essentiel d'insister pour que ces fonctions soient exercées de manière parfaite !

Le tuteur veillera à stimuler ceux qui prennent en charge les différentes fonctions, pas à prendre leur place …

Des fonctions pour faciliter le travail en équipe…

Pour que le travail en équipe se déroule bien et qu'il soit efficace, un peu d'**organisation** est nécessaire… Le tuteur vous aura remis des fiches/cartes qui décrivent différentes fonctions à assumer pour atteindre cet objectif.

Le verso de chaque carte précise en quoi consiste la fonction définie par la carte. Examinez les cartes et répartissez les fonctions entre les membres. Chacun dispose devant lui (ou elle !) la/les carte(s) qui lui est/sont attribuée(s) de façon à ce que chaque membre puisse voir qui prend en charge quelle(s) fonction(s).

Parmi les fonctions proposées, la fonction « **Participant actif** » doit être assumée par chacun des membres !

Quelques fonctions à répartir :

Fonctions indispensables :

Barreur 	Vous veillez à l'avancement du travail. Vous faites en sorte que l'équipe suive les étapes imposées ou qu'elle a décidé de suivre. Vous évitez que l'équipe se fourvoie, perde du temps dans des pistes sans issue.
Activateur 	Vous amenez chaque membre de l'équipe à contribuer activement aux travaux ; vous n'oubliez ni le scribe, ni le secrétaire ! En cas de nécessité de répartition de tâches, vous veillez à ce que chaque membre contribue de manière équitable.
Gardien du temps 	Vous veillez à la bonne utilisation du temps disponible. Vous attirez l'attention sur le risque de prendre du retard.
Scribe 	Sur l'espace de travail commun (p. ex. : flip chart), vous notez les idées importantes, les questions en suspens, les schémas qui émergent lors des discussions, mais sans imposer vos propres points de vue. Vous gérez les feuilles du flip chart pour que l'information utile soit visible pour tous les membres de l'équipe. Vous n'oubliez pas de participer aux discussions !
Secrétaire 	Vous produisez une synthèse des éléments importants issus des discussions : ceux qu'il faut conserver pour la suite du travail. Vous consignez toutes les informations nécessaires à la poursuite du travail : les décisions prises, les échéances déterminées, les prochains rendez-vous, les plans de travail collectifs et/ou individuels, etc. Vous diffusez vos productions et les autres documents nécessaires à l'ensemble des membres de l'équipe. Vous n'oubliez pas de participer aux discussions !

Fonctions pouvant être utiles :

Circulateur de parole 	Vous faites en sorte que chaque membre de l'équipe puisse s'exprimer. Vous incitez les membres en retrait à prendre la parole ; vous n'oubliez ni le scribe, ni le secrétaire ! Vous empêchez l'un ou l'autre membre de l'équipe de mobiliser la parole au détriment des autres.
Porte-parole 	Vous présentez l'état ou les résultats du travail de votre équipe d'une manière synthétique et complète, sans marquer de préférence pour votre propre point de vue. Vous utilisez tous les moyens nécessaires pour une communication efficace.
Faiseur de point 	Vous faites périodiquement le point sur l'état d'avancement : où en est l'équipe ? qu'est-ce qui est fait ? qu'est-ce qui reste à faire ? que savons-nous et que ne savons-nous pas ? Vous aidez le scribe à noter ces éléments sur l'espace de travail commun.
…	*Le cas échéant, ajoutez une fonction qui vous semble utile ou nécessaire*