En plus d’implémenter votre agent, vous devez rédiger un rapport qui détaille votre stratégie de recherche et vos choix de conception. Celui-ci peut être rédigé au choix en français ou en anglais et rendu au format PDF. Étant donné que vous êtes libres d’implémenter la solution de votre choix, qu’elle soit inspirée de concepts vus en cours, ou de vos connaissances personnelles, nous attendons une explication claire et détaillée du fonctionnement de votre agent. Précisément, votre rapport doit contenir au minimum les informations suivantes :   
  
1. Titre du projet.  
  
 2. Nom d’équipe sur Challonge, ainsi que la liste des membres de l’équipe (nom complet et matricule).   
  
Ireina Hedad 2213673  
  
  
3. Méthodologie : Explication du fonctionnement de votre agent, des choix de conception faits (fonctionnement de l’heuristique utilisée, stratégie choisie, spécificités propres de votre agent, gestion des 15 minutes allouées, etc.).  
  
**MCTS au debut  
min & max lorsque peu d’action possible  
alphabeta pruning**  
  
La structure de notre agent utilise le monte Carlo Tree search en début de partie puis dévie vers la stratégie du Minimax. Pour se faire, on detecte à quel stade de la partie avec l’attribut is\_early\_game qui est vrai lorsque nous somme dans les 10 premieres actions ou quand le plateau est occupé à 20% ou moins. Pour la gestion du temps, nous avons fixer le nombre d’itération de mtcs ainsi que la profondeur d’exploration de Minimax. L’attribut mtcs\_iterations a donc été fixé à xxx, ce qui signifie que l’on visite xxxx   
et l’attribut minimax\_depth à 3, donc on se rend 3 niveau plus bas avant de remonter la valeur. Les valeurs ont été choisie de sorte à ce que le temps total d’une partie ne dépasse jamais le 15 minutes allouées.  
  
Donc tout d’abord la fonction compute action est appelée, celle ci réduit le nombre d’action possible à 40 puisqu’en début de partie, le nombre d’action est très élévé et diminue à mesure que la partie avance. De plus, le choix d’action prend de l’importance à mesure que la partie avance. Pour choisir les 40 actions à garder, on utilise une heuristique ou pas. Ensuite, dependement de si on est en debut de partie ou non, on rentrera dans la fonction correspondante et la sortie de compute\_action sera la meilleure action renvoyée par find\_best\_action\_with\_mcts

ou find\_best\_action\_with\_minimax.

La fonction find\_best\_action\_with\_mcts va simuler des parties aléatoires pour choisir la meilleure action. Pour se faire, on utilise select\_action\_by\_ucb, une fonction qui retournera le meilleur coup à explorer parmi les nombreuses actions possibles en début de partie. Pour se faire, la fonction calcul le score UCB pour chacune des actions prises en paramètre et retourne celle avec le plus haut score. Le UCB score est l’addition du win\_rate qui favorise les coups qui gagnent le plus souvent et de exploration qui teste les coups peu joués pour éviter de rester bloqué.  
  
Après avoir récupéré cette action, on calcul le resultat de cette action en simulant une partie aléatoire jusqu’à la fin avec simulate\_random\_game. Cette fonction simule les prochaines actions des 2 joueurs purement aléatoirement et lorsque le résultat est renvoyé, find\_best\_action\_with\_mcts ajoute un nouveau noeud à son espace de recherche et met à jour la valeur de result avec celles renvoyé par simulate\_random\_game et ajoute 1 à la valeur de wins.

On refait ensuite le même calcul pour le nombre d’itération défini au début. Lorsque le nombre d’itération est atteint, on renvoie à compute\_action celle ayant le maximum win\_rate : Win\_rate = win/plays.  
  
REVOIR WIN RATE EXPLICATION

Pour ce qui est de Minimax, la fonction find\_best\_action\_with\_minimax

##### **3. Minimax Logic (**find\_best\_action\_with\_minimax**,**max\_value**,**min\_value**)**

**Alpha-Beta Pruning**: Eliminates 30-50% of branches

**Action Sorting**: Evaluates top 10(max)/5(min) actions first

**Depth Limit**: 3-ply search for responsiveness

##### **4. Evaluation Function**

**Heuristic**: Simple score difference (player\_score - opponent\_score)

**Design Choice**: Favors:

Immediate point gains

Blocking opponent scoring opportunities

**Améliorations possibles**

**Heuristique plus sophistiquée: prendre en compte la position des jetons, les motifs, etc.**

**Gestion adaptative de la profondeur: augmenter la profondeur quand il reste du temps**

**Parallelisation: explorer les branches de l'arbre en parallèle**

**Mémoire transposition: stocker les états déjà évalués pour éviter les recalculs**  
  
  
 4. Résultats et évolution de l’agent : Reportez les performances de votre agent en le testant contre les implémentations qui vous sont fournies et vos différentes versions. Spécifiez un ou plusieurs critères de mesure pour chiffrer vos résultats. Au-delà du simple taux de victoire ou score, mettez-les en avant à l’aide de graphiques, tableaux ou toutes autres techniques de visualisation pertinentes selon vous. Nous vous suggérons aussi d’utiliser vos parties sur Abyss pour appuyer vos analyses.   
  
5. Discussion : Discutez des avantages et limites de votre agent final. Évoquez différentes pistes d’améliorations envisageables selon vous.  
  
 6. Références (si applicable).  
  
 7. Annexes (si applicable).   
Le rapport ne doit pas dépasser 5 pages et doit être rédigé sur une ou deux colonnes à simple interligne, avec une police de caractère 10 ou plus (des pages supplémentaires pour les références et le contenu bibliographiques sont autorisées, ainsi que pour la page de garde). Si vos graphiques et tableaux vous font dépasser la limite, vous pouvez utiliser la section Annexes. Vous êtes libre de structurer le rapport comme vous le souhaitez du moment que tous les éléments mentionnés précédemment sont inclus