Les jeux - Introduction

- L'IA aime les jeux !
  - (Presque) Tous les hommes jouent, depuis toujours, quelque soit la civilisation.
  - Résultats faciles à vulgariser par les chercheurs.
- Les machines sont adaptées aux univers ludiques :
  - Loin des problèmes (physiques) du monde réel.
  - Tout est (souvent) formalisable.
  - Abstraction complète des problèmes.

Les jeux - Introduction

- Cas particulier dont nous ne traiterons pas : les robots.
- Trop de problèmes physiques, nous n'étudierons que les problèmes abstraits.

La RoboCup.

L'IA aime les jeux !

- Facile de mesurer expérimentalement les progrès dans un jeu à deux adversaires :
- Vitrine historique de l'IA :
  - Populaires pour démontrer les nouvelles idées en IA.
  - Résultats spectaculaires pour certains jeux: échecs, GO ...

L'IA aime les jeux !

- Facile de mesurer expérimentalement les progrès dans un jeu à deux adversaires :
- Vitrine historique de l'IA :
  - Populaires pour démontrer les nouvelles idées en IA.
  - Résultats spectaculaires pour certains jeux: échecs, GO ...

Les jeux aiment l'IA

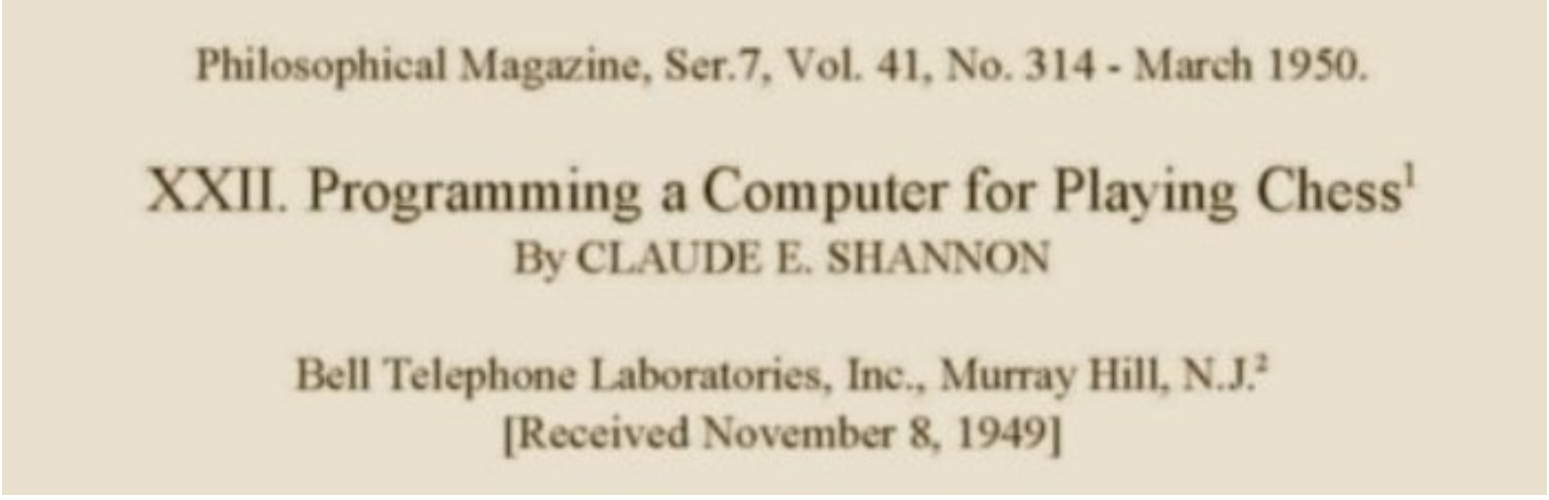
- Jeux Vidéos :
  - *On achète un jeu pour ses qualités graphiques mais on reste dessus pour ses qualités de jeux.*
- Des moyens énormes :
  - Les éditeurs de jeux font de plus en plus d'effort au niveau de l'IA.
  - La puissance de calcul des consoles est en partie dédiées au calcul pur, donc aux algorithmes de jeux.
- Reste des contraintes fortes comme le temps réel ou les implémentations de stratégies ...

Les jeux qui vont nous concerner

- On ne s'intéressera pas aux jeux physiques ou avec des contraintes temps réel fortes.
- Nous allons étudier :
  - Règles de jeu formalisables et précises.
  - Deux joueurs adversaires.
  - Jeux à somme zéro - ce que l'un perd, l'autre le gagne.
- On peut étendre les résultats aux jeux à plusieurs joueurs, avec tirage et connaissances imparfaites .... peut-être !!

Historique des jeux

- On considère que le premier article sur l'IA pour les jeux a été écrit par Shannon en 1950 à propos du jeu d'échecs.



Philosophical Magazine, Ser.7, Vol. 41, No. 314 - March 1950.

XXII. Programming a Computer for Playing Chess[1]

By CLAUDE E. SHANNON

Bell Telephone Laboratories, Inc., Murray Hill, N.J.[2]

[Received November 8, 1949]

Historique des jeux

- Article fondateur :

1. INTRODUCTION

This paper is concerned with the problem of constructing a computing routine or "program" for a modern general purpose computer which will enable it to play chess. Although perhaps of no practical importance, the question is of theoretical interest, and it is hoped that a satisfactory solution of this problem will act as a wedge in attacking other problems of a similar nature and of greater significance. Some possibilities in this direction are: -

(1)Machines for designing filters, equalizers, etc.
(2)Machines for designing relay and switching circuits.
(3)Machines which will handle routing of telephone calls based on the individual circumstances rather than by fixed patterns.
(4)Machines for performing symbolic (non-numerical) mathematical operations.
(5)Machines capable of translating from one language to another.
(6)Machines for making strategic decisions in simplified military operations.
(7)Machines capable of orchestrating a melody.
(8)Machines capable of logical deduction.

Historique des jeux

- Article fondateur :

1. INTRODUCTION

This paper is concerned with the problem of constructing a computing routine or "program" for a modern general purpose computer which will enable it to play chess. Although perhaps of no practical importance, the question is of theoretical interest, and it is hoped that a satisfactory solution of this problem will act as a wedge in attacking other problems of a similar nature and of greater significance. Some possibilities in this direction are: -

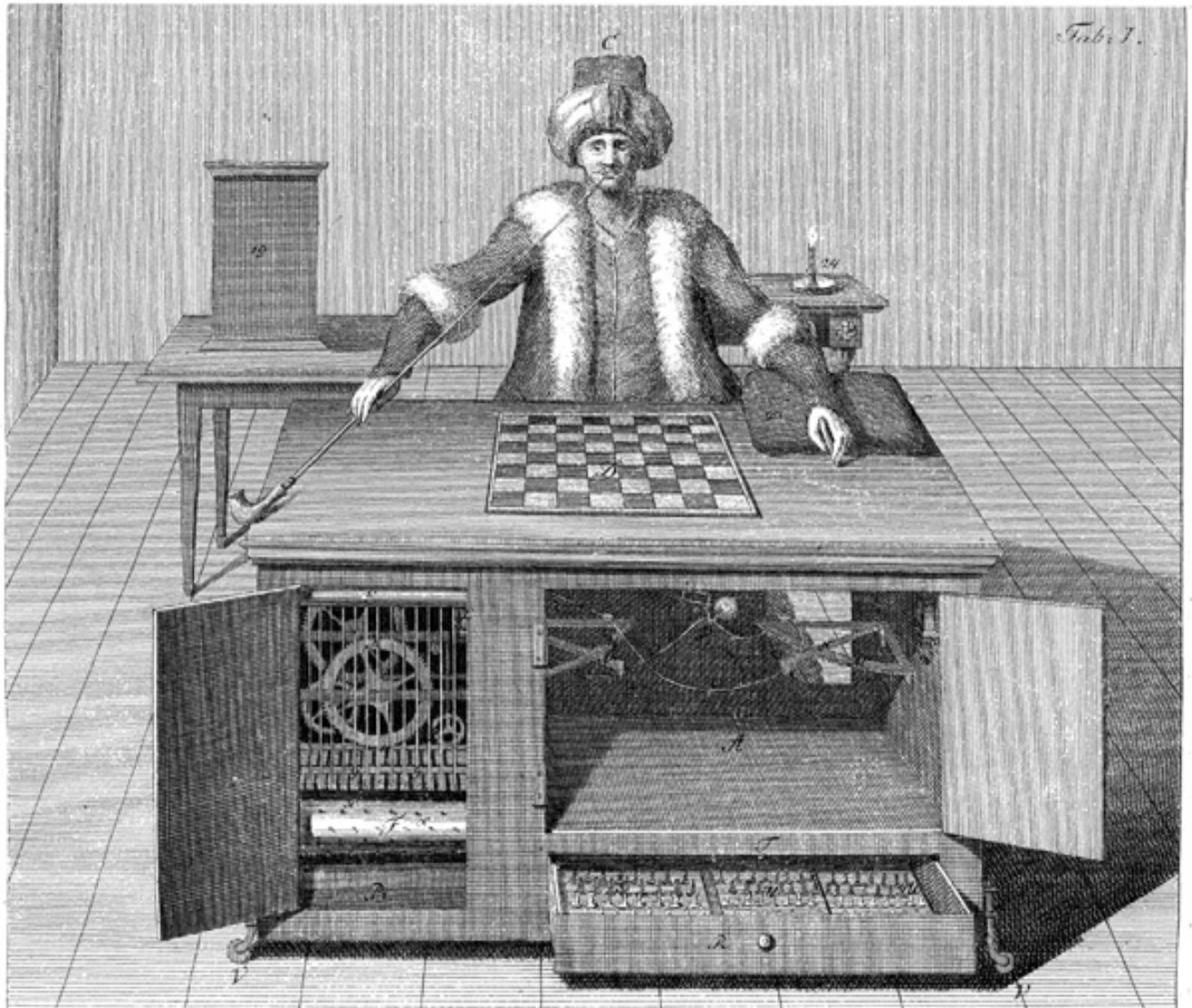
(1)Machines for designing filters, equalizers, etc.
(2)Machines for designing relay and switching circuits.
(3)Machines which will handle routing of telephone calls based on the individual circumstances rather than by fixed patterns.
(4)Machines for performing symbolic (non-numerical) mathematical operations.
(5)Machines capable of translating from one language to another.
(6)Machines for making strategic decisions in simplified military operations.
(7)Machines capable of orchestrating a melody.
(8)Machines capable of logical deduction.

Historique des jeux

- Mais avant cela !

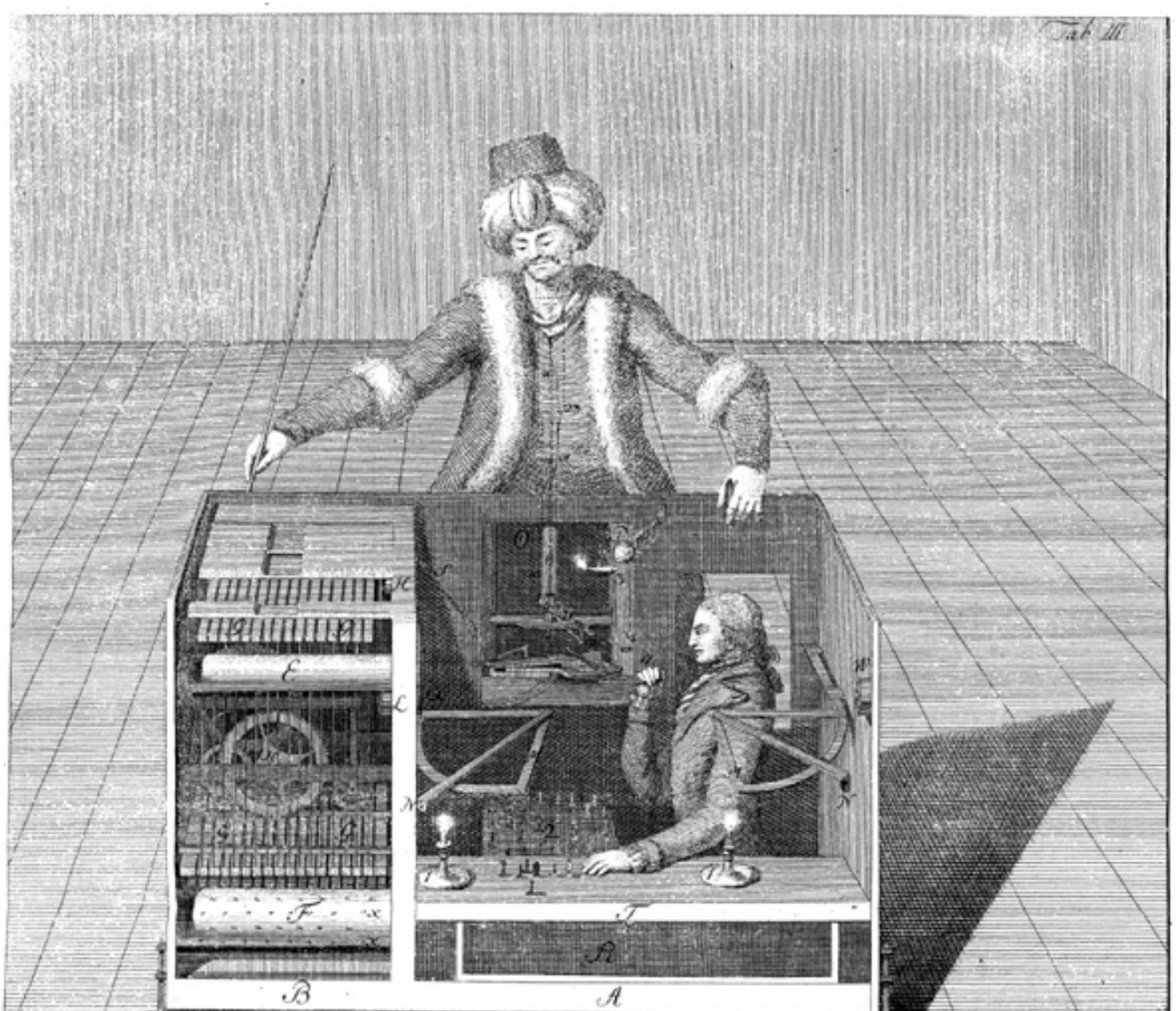
Historique des jeux

- Mais avant cela !
- Une machine qui joue aux échecs en 1769 ?



Historique des jeux

- Mais avant cela !
- Une machine qui joue aux échecs en 1769 ?



Historique des jeux

- On commence avec les échecs :

1945

1945 Turing présente les échecs comme ce que peuvent faire les ordinateurs

1946 Turing parle d'intelligence des machines, en relation avec les échecs

1950 Turing écrit le premier programme de jeux (la même année que le test de Turing)

1950 Shannon écrit le premier article scientifique sur les échecs

1951 Turing simule le premier programme d'échecs à la main contre un joueur assez mauvais. Il perd.

1952 A. Glennie est le premier humain à battre un ordinateur aux échecs (*TurboChamp*, de Turing)

1957 H. Simon prédit que d'ici 10 ans, le champion du monde d'échecs sera un ordinateur.

1957

Historique des jeux

Les premiers succès : les dames

- Premier programme à battre un champion du monde humain.
  - A. Samuel étudie les dames dès 1959.
  - En 1963 son programme gagne contre un champion.
  - en 1970 il est battu par un programme de la Duke University dans un match court.
  - Ils font partie des 10 meilleurs joueurs mondiaux.

Historique des jeux

Les premiers succès : les dames

- Premier programme à battre un champion du monde humain.
  - A. Samuel étudie les dames dès 1959.
  - En 1963 son programme gagne contre un champion.
  - en 1970 il est battu par un programme de la Duke University dans un match court.
  - Ils font partie des 10 meilleurs joueurs mondiaux.
  - En 1989 arrivée de *Chinook* (Schaffer), création de *The Man vs Machine World Championship*.
  - En 1992 *Chinook* perd Tinsley (2 contre 4 et 33 null) puis gagne en 1995 contre Don Lafferty.
  - En 1997 *Chinook* est retiré des compétitions humaines.
  - En juillet 2007 *Science* publie l'article de l'équipe de Schaeffer *"Checkers is solved"* qui démontre que le meilleur résultat possible si 2 joueurs jouent de facon optimale est une partie nulle.

Kasparov vs Deep Blue

- Deep Blue (IBM) en 1996 mesurait 2m et pesait 700kg.
- En 1996, Kasparov gagne avec 3 victoires contre 1 pour Deep Blue.
- Match revanche en 1997 : Deep Blue bat Kasparov avec 3 victoires contre 2. Deep(er) Blue pèse alors 1.4 tonnes et mesure 1.80m. Il pouvait calculer 300 millions de position/seconde.

Lee Sedol vs AlphaGo

- Match de 5 parties du jeu de Go.
- Le nombre de combinaisons du jeu de Go sont beaucoup plus nombreuses qu'aux échecs.
- Lee Sedol est un champion coréen, joueur professionel.
- AlphoGo a été développé par Google DeepMind.
- Matche en 2016 gagné par AlphaGo, c'est la 1iére victoire d'un programme sur un expert humain du jeu de Go.
- Certains joueurs ont déclaré avoir appris du jeu réalisé par AlphaGo.

Lee Sedol vs AlphaGo

- Match de 5 parties du jeu de Go.
- Le nombre de combinaisons du jeu de Go sont beaucoup plus nombreuses qu'aux échecs.
- Lee Sedol est un champion coréen, joueur professionel.
- AlphoGo a été développé par Google DeepMind.
- Matche en 2016 gagné par AlphaGo, c'est la 1iére victoire d'un programme sur un expert humain du jeu de Go.
- Certains joueurs ont déclaré avoir appris du jeu réalisé par AlphaGo.
- C'est le premier programme qui utilise l'algorithme `MinMax` et des réseaux de neurones pour faire évoluer les règles de comportement.

Les algorithmes de jeux

- Recherche totale :
  - Simulation de tous les coups possibles depuis la position initiale permet de construire l'arbre de jeu.
  - Exploration de l'arbre de jeu.

Algorithme MinMax

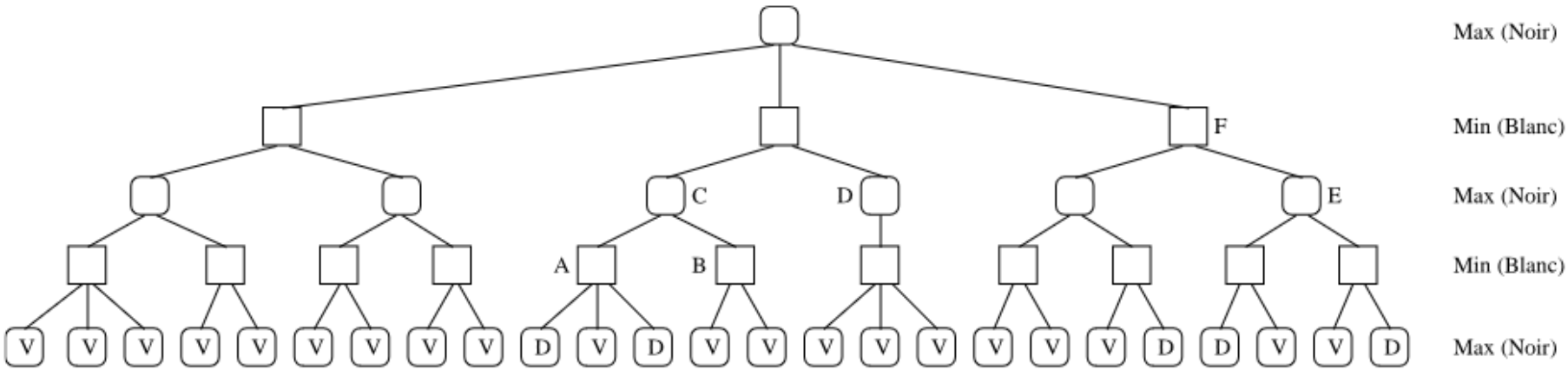
- Vient de la théorie des jeux de John Von Neumann (1926).
- Appelé “Théorème fondamental de la théorie des jeux à deux joueurs et à somme nulle.”
- Jeux à 2 joueurs et à somme nulle.
- Connaissances complètes des possibilités d’action de l’opposant.
- L’amélioration la plus connue est celle faisant intervenir l’élagage : AlphaBeta.

Algorithme MinMax

- Soit 2 joueurs : **Noir** et **Blanc** (comme dans le jeu d'Othello).
- On considère la configuration où **Noir** veut MAXimiser ses gains.
- Par conséquent **Blanc** tentera de MINimiser les gains de **Noir** afin de gagner (jeu à somme nulle).
- Construction de l'arbre de jeu.

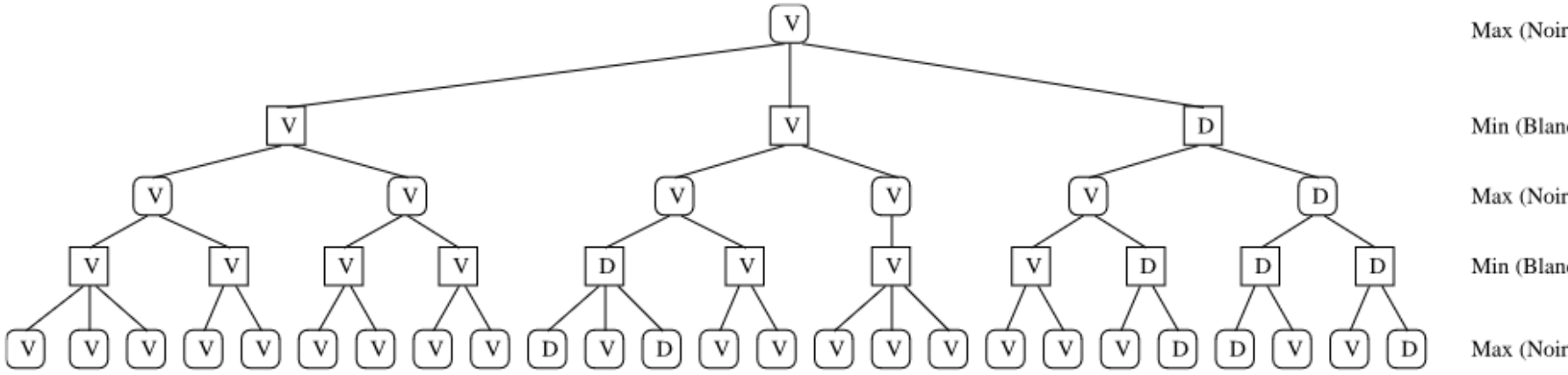
Algorithme MinMax

- Valuer les feuilles avec le résultat final.



Algorithme MinMax

- Parcours de l'arbre pour étiqueter les noeuds et faire remonter l'information issue des feuilles.



Algorithme MinMax

- Pseudo code :

```
int fonction minimax (int profondeur)
{
   .....
   if (noeud == MAX) { //=Programme
      meilleur_score = -INFINITY;
      for (chaque coup possible m) {
            jouer le coup m
            int score = minimax (profondeur - 1)
            annuler le coup m;
            if (score > meilleur_score) {
               meilleur_score = score;
               meilleur_coup = m ;
            }
      }
}
```

Algorithme MinMax

- Pseudo code :

```
else { //type MIN = adversaire
  meilleur_coup = +INFINITY;
  for (chaque coup possible m) {
    jouer le coup m;
    int score = minimax (profondeur - 1)
    annuler le coup m;
    if (score < meilleur_score) {
      meilleur_score = score;
      meilleur_coup = m ;
    }
  }
}
```

Algorithme MinMax

- Pseudo code :

```
int fonction minimax (int profondeur)
{
  if (game_over or profondeur = 0)
  return evaluation();
  ...../// le cas Max

  ..../// le cas Min

 return meilleur_coup ;
}
```

Algorithme MinMax

- Ce pseudo-code n'est pas très optimisé !

Algorithme MinMax

- Ce pseudo-code n'est pas très optimisé !
- On peut observer les grandes similitudes entre le code pour les deux cas.

Algorithme MinMax

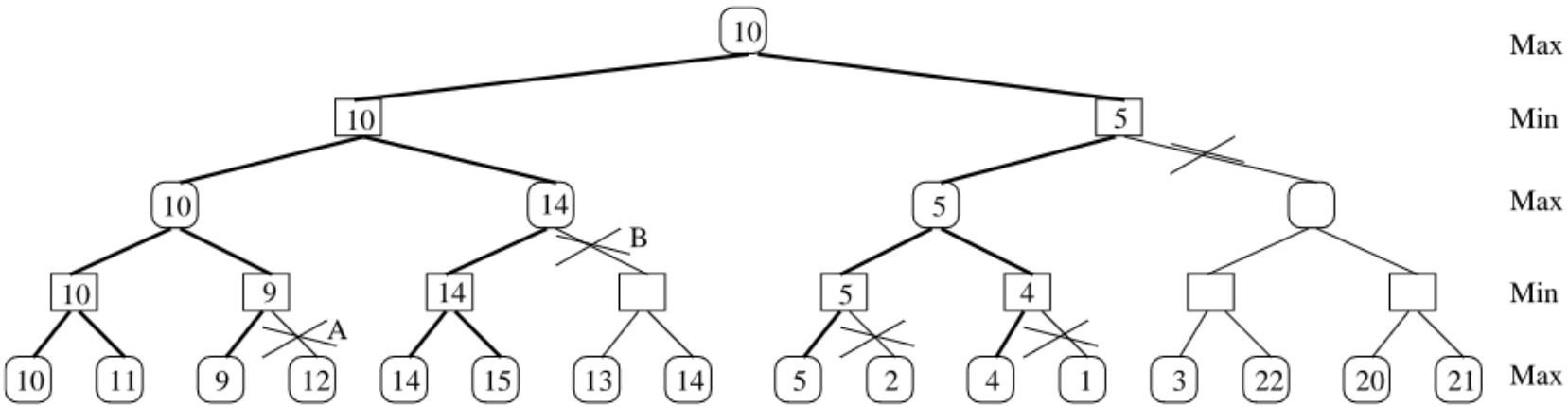
- Ce pseudo-code n'est pas très optimisé !
- On peut observer les grandes similitudes entre le code pour les deux cas.
- Les limites : le nombre de branches qui sont développées.

Algorithme MinMax

- Ce pseudo-code n'est pas très optimisé !
- On peut observer les grandes similitudes entre le code pour les deux cas.
- Les limites : le nombre de branches qui sont développées.
- Existe-t-il une solution ?

Elagage de l'arbre

- Principe : Eviter la génération de branches et de feuilles inutiles.



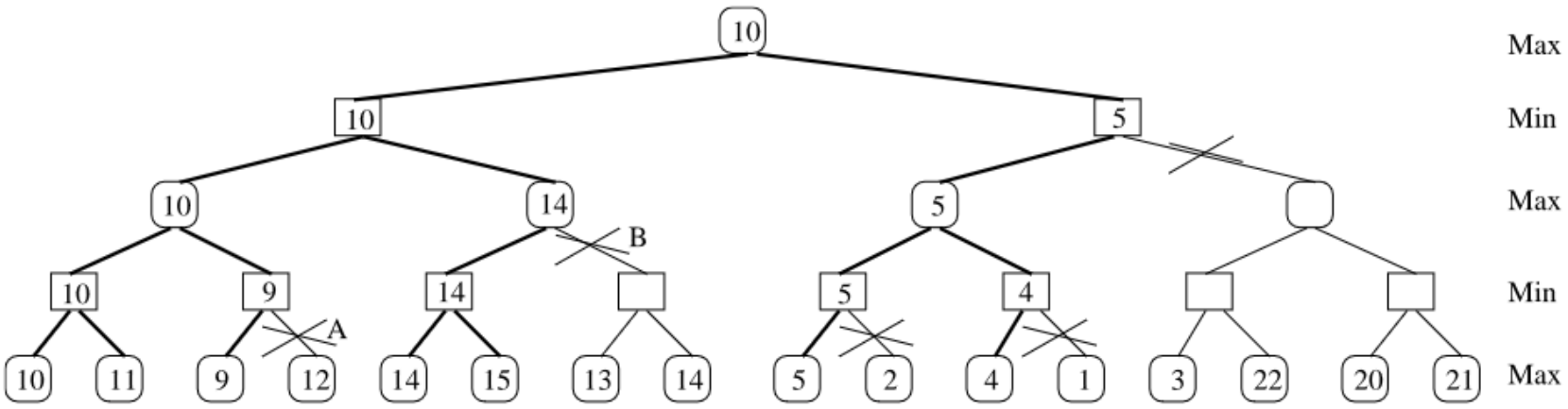
AlphaBeta

- Utilisation du parcours en profondeur : avant d'examiner les frères d'un noeud on développe ses fils.
- Méthode : mettre à jour deux variables α et β qui contiennent respectivement la valeur minimale que le joueur peut espérer avec ce coup et la valeur maximale.

AlphaBeta

- Utilisation du parcours en profondeur : avant d'examiner les frères d'un noeud on développe ses fils.
- Méthode : mettre à jour deux variables α et β qui contiennent respectivement la valeur minimale que le joueur peut espérer avec ce coup et la valeur maximale.
- La construction de certaines branches sera alors arrêtée si elles contiennent la possibilité pour un joueur de faire des coups qui ne respectent pas la règle que α est la note la plus basse que peut obtenir le joueur **MAX** ou que β est la valeur maximale que le joueur **MIN** autorisera **MAX** à obtenir.
- En pratique on fixe aussi la profondeur à laquelle on veut examiner le jeu.

Elagage de l'arbre



AlphaBeta

- Permet la suppression de grande partie de l'arbre complet.
- Induit une réduction substantielle des temps de calcul de la décision de jeu.
- Un arbre élagé par AlphaBeta renvoie le même résultat que MinMax.

AlphaBeta

- Permet la suppression de grande partie de l'arbre complet.
- Induit une réduction substantielle des temps de calcul de la décision de jeu.
- Un arbre élagé par AlphaBeta renvoie le même résultat que MinMax.

Autres Optimisations :

- Inutile en début de partie, entre autre à cause du choix de la profondeur.
- Stratégie classique : calculer à l'avance les premiers coups avec une profondeur élévée, identique à l'apprentissage des coups d'ouverture.
- Le choix de la procédure peut varier en fonction du moment de la partie.