Intelligence Artificielle

– jeux de plateaux (1)

Laurent SIMON
Bordeaux-INP / LaBRI
@lorensipro
lsimon@labri.fr

L'I.A. aime les jeux !

Preuve d'*intelligence* ?

- (Presque) Tous les hommes jouent
- Certains jeux ont un lourd passé humain
- 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

Robocup 2005, trop de problèmes physiques ?
Dans ce cours on n'étudie *que* les problèmes *abstraits*.

L'I.A. aime les jeux !

Preuve d'*intelligence* ?

- (Presque) Tous les hommes jouent
- Certains jeux ont un lourd passé humain
- 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

Robocup 2005, trop de problèmes physiques ?
Dans ce cours on n'étudie *que* les problèmes *abstraits*.

L'I.A. aime les jeux !

Preuve d'*intelligence* ?

- (Presque) Tous les hommes jouent
- Certains jeux ont un lourd passé humain
- 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

Robocup 2005, trop de problèmes physiques ?
Dans ce cours on n'étudie *que* les problèmes *abstraits*.

L'I.A. aime les jeux !

Preuve d'*intelligence* ?

- (Presque) Tous les hommes jouent
- Certains jeux ont un lourd passé humain
- 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

Robocup 2005, trop de problèmes physiques ?
Dans ce cours on n'étudie *que* les problèmes *abstraits*.

L'I.A. aime les jeux !

Preuve d'*intelligence* ?

- (Presque) Tous les hommes jouent
- Certains jeux ont un lourd passé humain
- 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

Robocup 2005, trop de problèmes physiques ?
Dans ce cours on n'étudie *que* les problèmes *abstraits*.

L'I.A. aime les jeux !

Preuve d'*intelligence* ?

- (Presque) Tous les hommes jouent
- Certains jeux ont un lourd passé humain
- 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

Robocup 2005, trop de problèmes physiques ?
Dans ce cours on n'étudie *que* les problèmes *abstraits*.

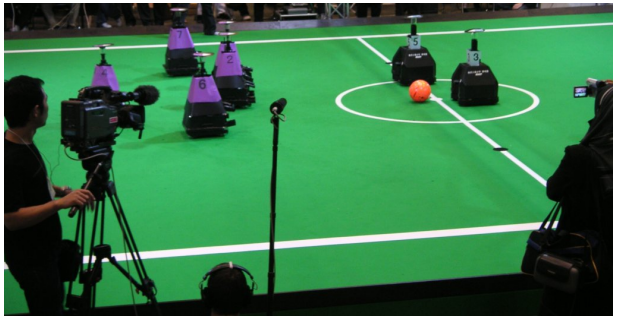
L'I.A. aime les jeux !

Preuve d'*intelligence* ?

- (Presque) Tous les hommes jouent
- Certains jeux ont un lourd passé humain
- 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



Robocup 2005, trop de problèmes physiques ?
Dans ce cours on n'étudie *que* les problèmes *abstraits*.

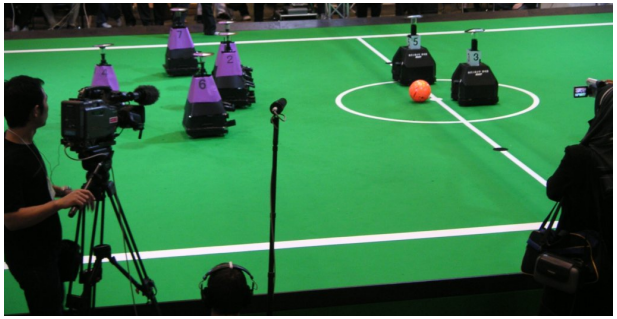
L'I.A. aime les jeux !

Preuve d'*intelligence* ?

- (Presque) Tous les hommes jouent
- Certains jeux ont un lourd passé humain
- 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



Robocup 2005, trop de problèmes physiques ?
Dans ce cours on n'étudie *que* les problèmes *abstraits*.

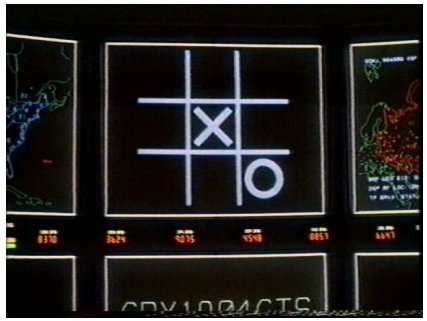
L'I.A. aime les jeux !

Et aussi...

Facile de mesurer *expérimentalement* les progrès dans un jeu à **deux adversaires**

Vitrine historique de l'I.A.

- Populaires pour démontrer les nouvelles idées en I.A.
- Quête Historique de l'I.A. (et dans le futur ?)
- Les résultats sont spectaculaires dans certains jeux
- Les techniques découvertes doivent servir à d'autres domaines de l'I.A.



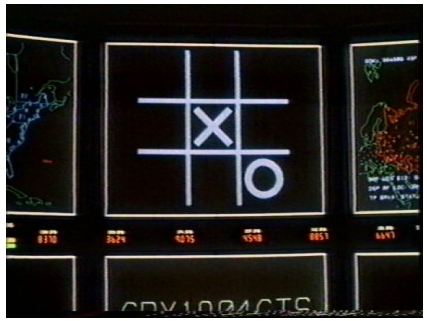
L'I.A. aime les jeux !

Et aussi...

Facile de mesurer *expérimentalement* les progrès dans un jeu à **deux adversaires**

Vitrine historique de l'I.A.

- Populaires pour démontrer les nouvelles idées en I.A.
- Quête Historique de l'I.A. (et dans le futur ?)
- Les résultats sont spectaculaires dans certains jeux
- Les techniques découvertes doivent servir à d'autres domaines de l'I.A.



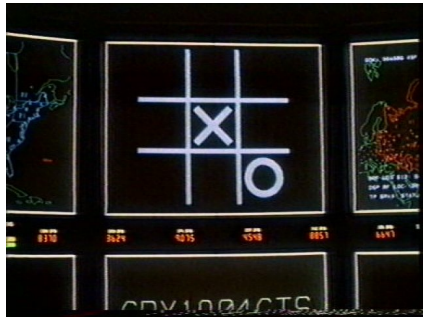
L'I.A. aime les jeux !

Et aussi...

Facile de mesurer *expérimentalement* les progrès dans un jeu à **deux adversaires**

Vitrine historique de l'I.A.

- Populaires pour démontrer les nouvelles idées en I.A.
- Quête Historique de l'I.A. (et dans le futur ?)
- Les résultats sont spectaculaires dans certains jeux
- Les techniques découvertes doivent servir à d'autres domaines de l'I.A.



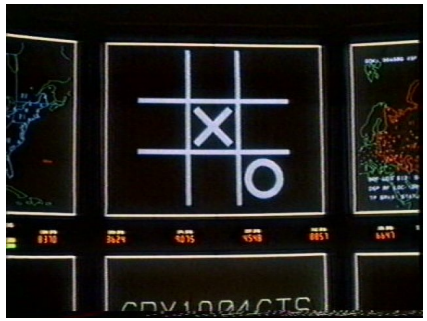
L'I.A. aime les jeux !

Et aussi...

Facile de mesurer *expérimentalement* les progrès dans un jeu à **deux adversaires**

Vitrine historique de l'I.A.

- Populaires pour démontrer les nouvelles idées en I.A.
- Quête Historique de l'I.A. (et dans le futur ?)
- Les résultats sont spectaculaires dans certains jeux
- Les techniques découvertes doivent servir à d'autres domaines de l'I.A.



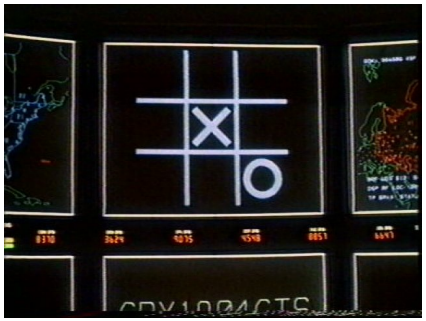
L'I.A. aime les jeux !

Et aussi...

Facile de mesurer *expérimentalement* les progrès dans un jeu à **deux adversaires**

Vitrine historique de l'I.A.

- Populaires pour démontrer les nouvelles idées en I.A.
- Quête Historique de l'I.A. (et dans le futur ?)
- Les résultats sont spectaculaires dans certains jeux
- Les techniques découvertes doivent servir à d'autres domaines de l'I.A.



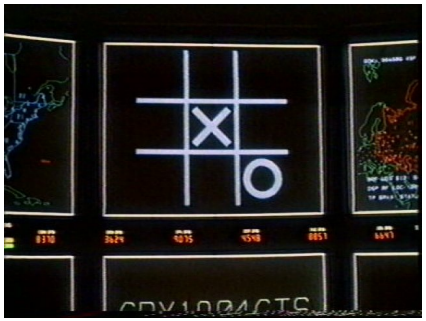
L'I.A. aime les jeux !

Et aussi...

Facile de mesurer *expérimentalement* les progrès dans un jeu à **deux adversaires**

Vitrine historique de l'I.A.

- Populaires pour démontrer les nouvelles idées en I.A.
- Quête Historique de l'I.A. (et dans le futur ?)
- Les résultats sont spectaculaires dans certains jeux
- Les techniques découvertes doivent servir à d'autres domaines de l'I.A.



Un film qui aime l'I.A. qui aime les jeux !

Les jeux aiment l'I.A.

Les jeux vidéos

« On achète un jeu pour ses qualités graphiques, mais on reste dessus pour ses qualités de jeux.»

De l'argent à profusion ?

Les éditeurs de jeux font de plus en plus d'effort au niveau de l'I.A. La puissance de calcul des consoles (PS4, XOne, Nintendo Switch) est en partie dédiées au calcul pur, donc aux I.A. des jeux.

Mais, comme pour les robots, c'est un *autre problème*

- Contrainte temps-réel forte
- Triche pour rendre les I.A. réalistes
- ...

Les jeux aiment l'I.A.

Les jeux vidéos

« On achète un jeu pour ses qualités graphiques, mais on reste dessus pour ses qualités de jeux.»

De l'argent à profusion ?

Les éditeurs de jeux font de plus en plus d'effort au niveau de l'I.A. La puissance de calcul des consoles (PS4, XOne, Nintendo Switch) est en partie dédiées au calcul pur, donc aux I.A. des jeux.

Mais, comme pour les robots, c'est un *autre problème*

- Contrainte temps-réel forte
- Triche pour rendre les I.A. réalistes
- ...

Les jeux aiment l'I.A.

Les jeux vidéos

« On achète un jeu pour ses qualités graphiques, mais on reste dessus pour ses qualités de jeux.»

De l'argent à profusion ?

Les éditeurs de jeux font de plus en plus d'effort au niveau de l'I.A. La puissance de calcul des consoles (PS4, XOne, Nintendo Switch) est en partie dédiées au calcul pur, donc aux I.A. des jeux.

Mais, comme pour les robots, c'est un *autre problème*

- Contrainte temps-réel forte
- Tricherie pour rendre les I.A. *réalistes*
- ...

Les jeux aiment l'I.A.

Les jeux vidéos

« On achète un jeu pour ses qualités graphiques, mais on reste dessus pour ses qualités de jeux.»

De l'argent à profusion ?

Les éditeurs de jeux font de plus en plus d'effort au niveau de l'I.A. La puissance de calcul des consoles (PS4, XOne, Nintendo Switch) est en partie dédiées au calcul pur, donc aux I.A. des jeux.

Mais, comme pour les robots, c'est un *autre problème*

- Contrainte temps-réel forte
- Tricherie pour rendre les I.A. *réalistes*
- ...

Les jeux aiment l'I.A.

Les jeux vidéos

« On achète un jeu pour ses qualités graphiques, mais on reste dessus pour ses qualités de jeux.»

De l'argent à profusion ?

Les éditeurs de jeux font de plus en plus d'effort au niveau de l'I.A. La puissance de calcul des consoles (PS4, XOne, Nintendo Switch) est en partie dédiées au calcul pur, donc aux I.A. des jeux.

Mais, comme pour les robots, c'est un *autre problème*

- Contrainte temps-réel forte
- Tricherie pour rendre les I.A. *réalistes*
- ...

Les jeux aiment l'I.A.

Les jeux vidéos

« On achète un jeu pour ses qualités graphiques, mais on reste dessus pour ses qualités de jeux.»

De l'argent à profusion ?

Les éditeurs de jeux font de plus en plus d'effort au niveau de l'I.A. La puissance de calcul des consoles (PS4, XOne, Nintendo Switch) est en partie dédiées au calcul pur, donc aux I.A. des jeux.

Mais, comme pour les robots, c'est un *autre problème*

- Contrainte temps-réel forte
- Tricherie pour rendre les I.A. *réalistes*
- ...

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Les jeux que l'on aime dans ce cours

On ne va pas étudier les jeux physiques ou avec contraintes temps réel forte.

Ce que l'on va étudier

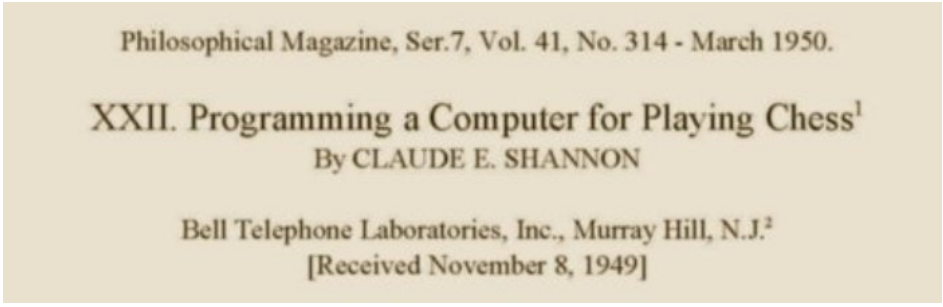
- Règles de jeu formalisables et précises
- Deux joueurs adversaires
- Jeux à somme zéro (ce que l'un perd, l'autre le gagne)
- Jeux ouverts

Les résultats pourront être étendus aux jeux :

- À plusieurs joueurs (mais toujours à somme zéro)
- Avec tirage de dés (ou de lettres)
- Avec contrainte temps réel (Ah ?)
- Avec connaissance imparfaite (Ah Ah ?)

Historique des jeux

Article fondateur de l'I.A.


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

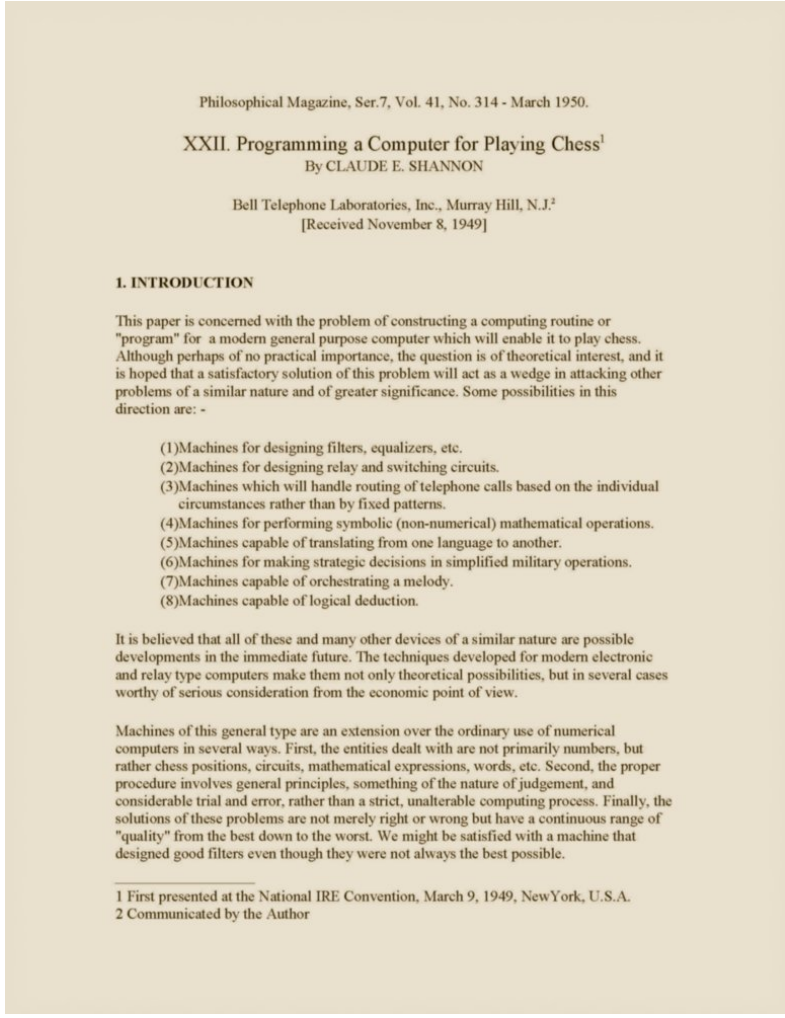
XXII. Programming a Computer for Playing Chess[1]

By CLAUDE E. SHANNON

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

[Received November 8, 1949]

Le **premier article d'I.A.** est un article sur les jeux (Shannon, 1950).


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]

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.

It is believed that all of these and many other devices of a similar nature are possible developments in the immediate future. The techniques developed for modern electronic and relay type computers make them not only theoretical possibilities, but in several cases worthy of serious consideration from the economic point of view.

Machines of this general type are an extension over the ordinary use of numerical computers in several ways. First, the entities dealt with are not primarily numbers, but rather chess positions, circuits, mathematical expressions, words, etc. Second, the proper procedure involves general principles, something of the nature of judgement, and considerable trial and error, rather than a strict, unalterable computing process. Finally, the solutions of these problems are not merely right or wrong but have a continuous range of "quality" from the best down to the worst. We might be satisfied with a machine that designed good filters even though they were not always the best possible.

1 First presented at the National IRE Convention, March 9, 1949, NewYork, U.S.A.
2 Communicated by the Author

Historique des jeux

Article fondateur de l'I.A.

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 de l'I.A.

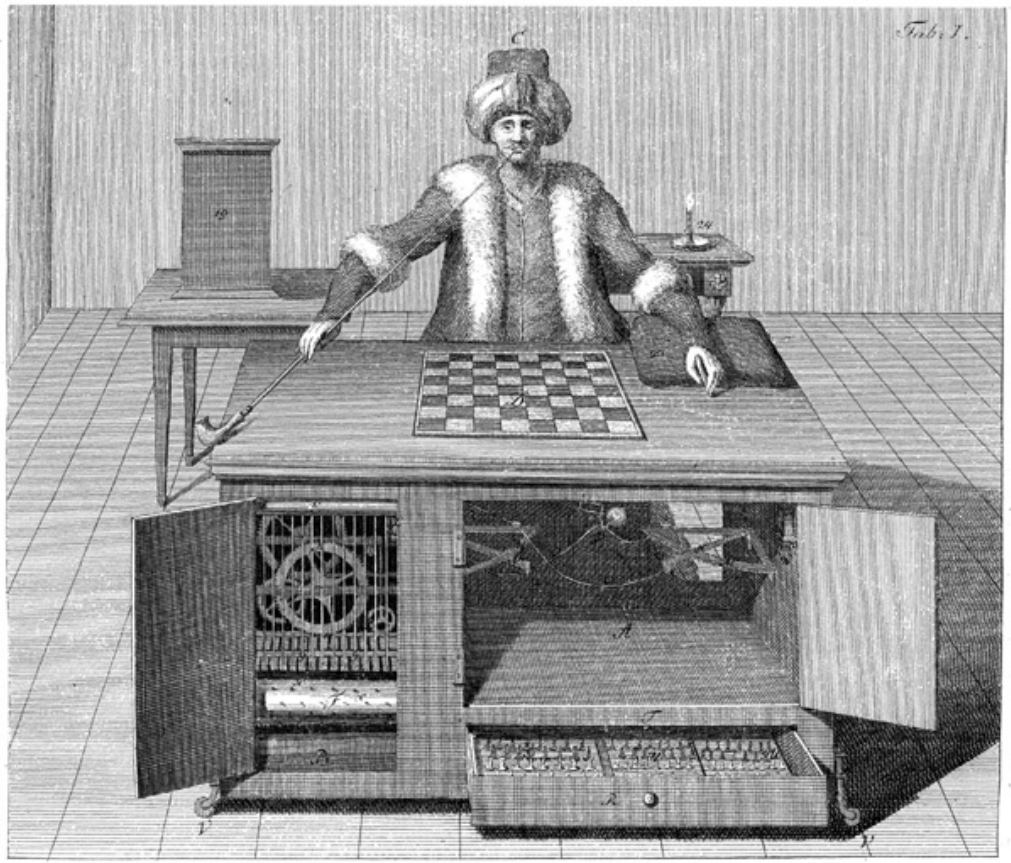
1. INTRODUCTION

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.

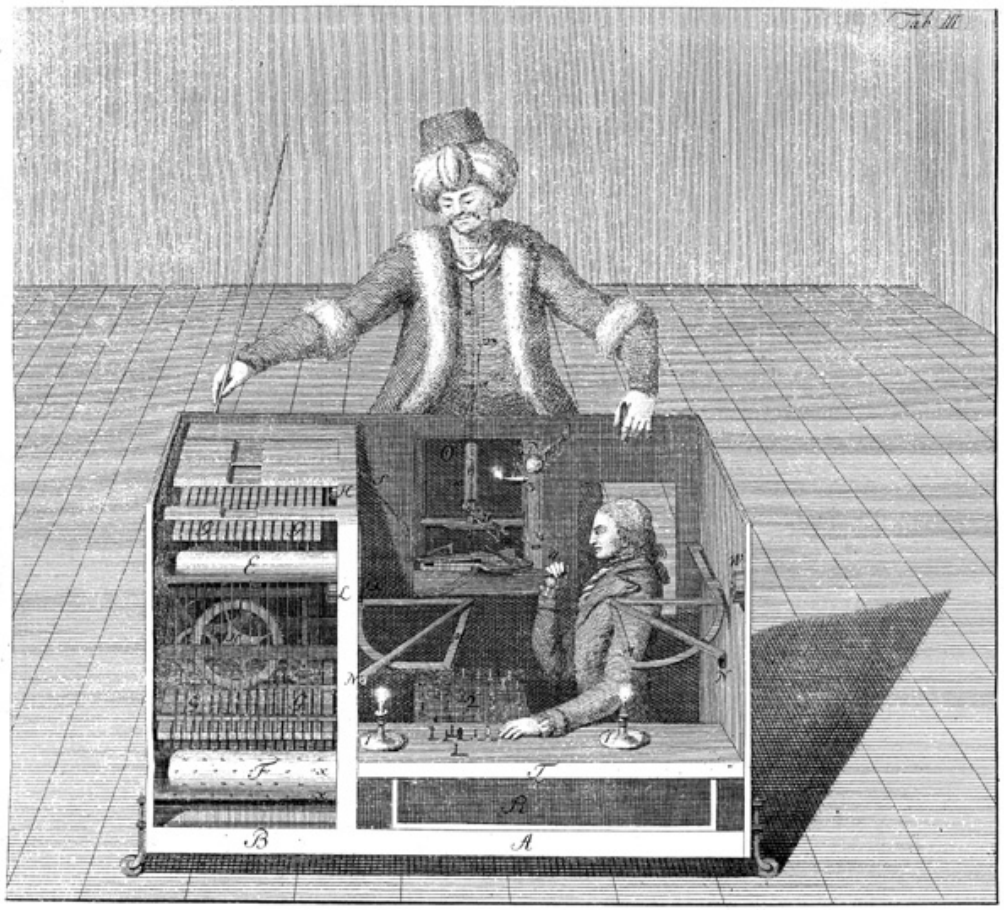
Historique des jeux

Une machine qui joue aux Échecs en 1769 ?



Historique des jeux

Une machine qui joue aux Échecs en 1769 ?



Historique des jeux

Une machine d'aujourd'hui qui joue aux Échecs comme le Turc ?

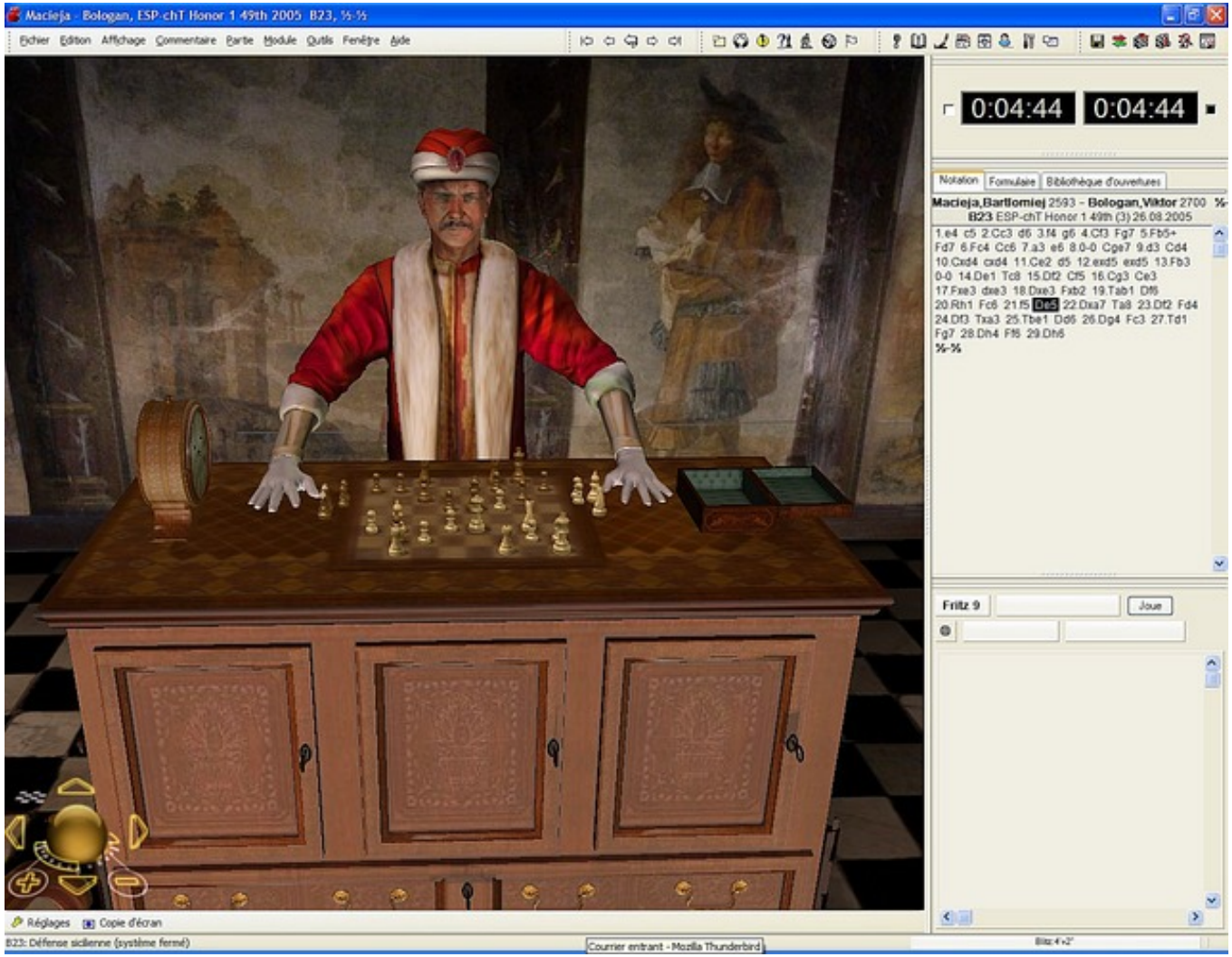


Image du *Turc mécanique* issue de Fritz 9 (2006).

Historique des jeux

On commence par des 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 jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. The World Man-Machine Championship est créé.
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. The World Man-Machine Championship en 1992.
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. The World Man-Machine Championship est créé.
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. *The World Man-Machine Championship* est crée
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. *The World Man-Machine Championship* est crée
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. *The World Man-Machine Championship* est crée
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. *The World Man-Machine Championship* est crée
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. *The World Man-Machine Championship* est crée
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. *The World Man-Machine Championship* est crée
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Historique des jeux

Les premiers succès : Les dames

Premier jeu 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 parti des 10 meilleurs joueurs mondiaux
- En 1989, des intérêts commerciaux entrent en jeu et *Chinook* [Schaeffer 1989] arrive. *The World Man-Machine Championship* est crée
- Après avoir perdu en 1992, *Chinook* gagne en 1994.
- *Chinook* n'a jamais plus perdu un match.
- Il a été retiré des compétitions humaines en 1997.

Aujourd'hui, **les Dames sont résolues.** (voir site web de Schaeffer sur Chinook)

Le domaine des machines

Jeux dans lesquels les ordinateurs sont bien supérieurs aux humains :

- Dames
- Othello
- Scrabble

Jeux dans lesquels les ordinateurs sont de niveau mondial :

- Échecs
- Backgammon (avec un peu de chance)
- Scrabble

Jeux encore challenging pour les programmes

- Go
- Bridge
- OCTI

Le domaine des machines

Jeux dans lesquels les ordinateurs sont bien supérieurs aux humains :

- Dames
- Othello
- Scrabble

Jeux dans lesquels les ordinateurs sont de niveau mondial :

- Échecs
- Backgammon (avec un peu de chance)
- Scrabble

Jeux encore challenging pour les programmes

- Go
- Bridge
- OCTI

Le domaine des machines

Jeux dans lesquels les ordinateurs sont bien supérieurs aux humains :

- Dames
- Othello
- Scrabble

Jeux dans lesquels les ordinateurs sont de niveau mondial :

- Échecs
- Backgammon (avec un peu de chance)
- Scrabble

Jeux encore challenging pour les programmes

- Go
- Bridge
- OCTI

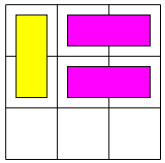
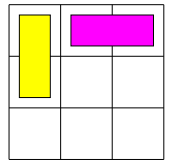
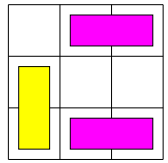
recherche totale

Un problème simple

Les dominos de couleurs

Principe

Sur une grille $n \times n$, on doit être le dernier à poser son domino 2×1.
Noir joue horizontalement et blanc verticalement.

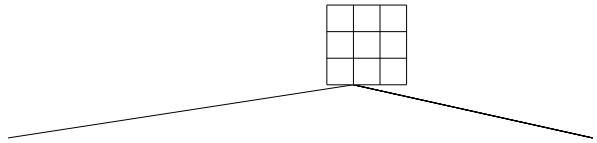




Un jeu simple mais illustratif

Nous allons utiliser ce jeu simple pour schématiser les approches

Exploration de l'arbre de jeu

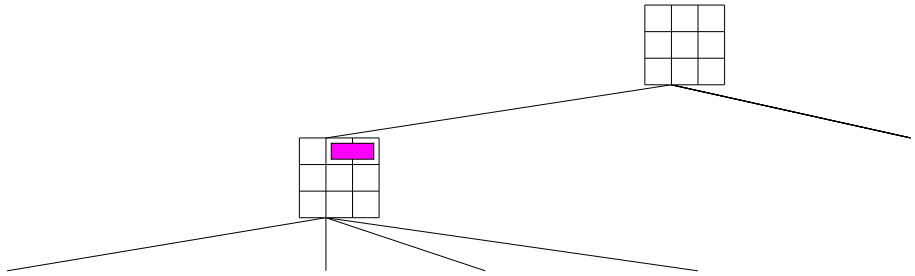
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

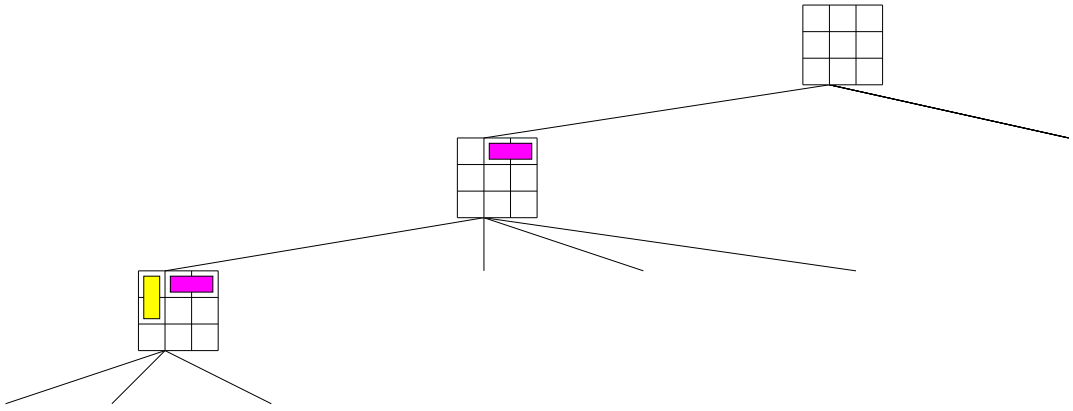
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

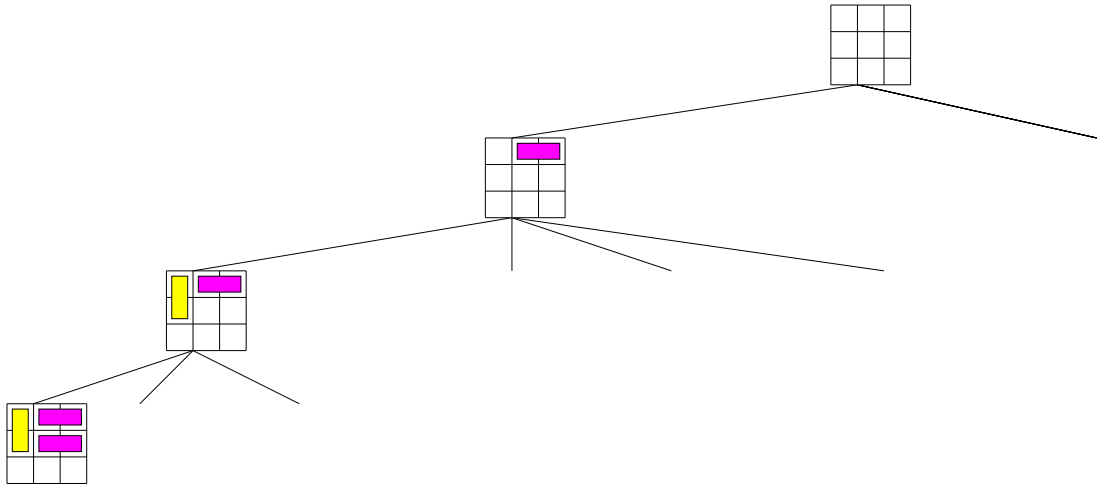
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

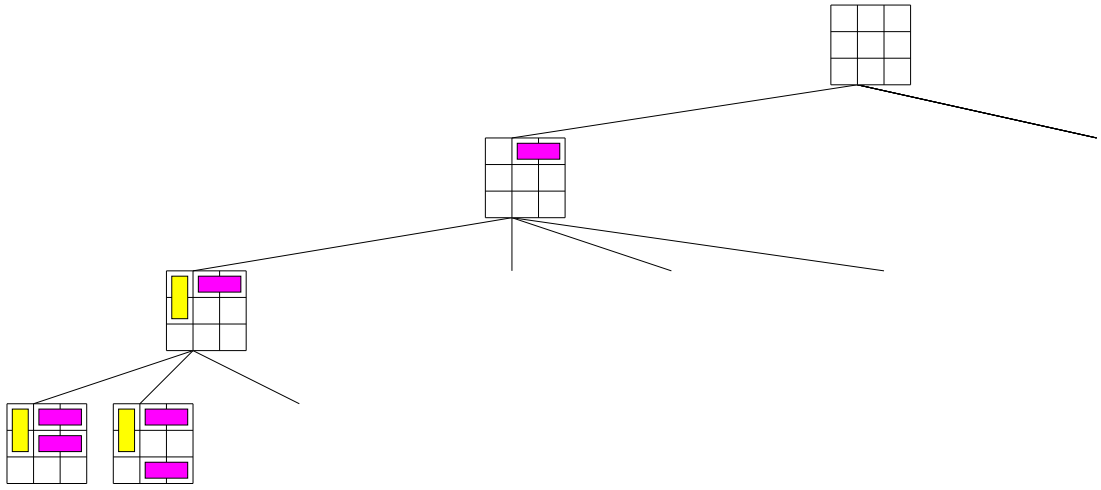
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

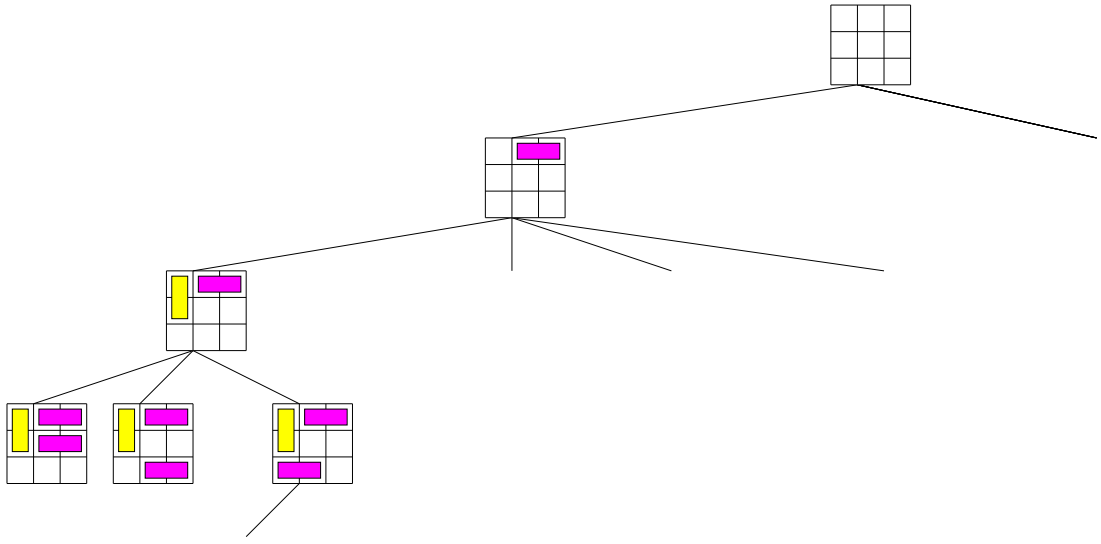
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

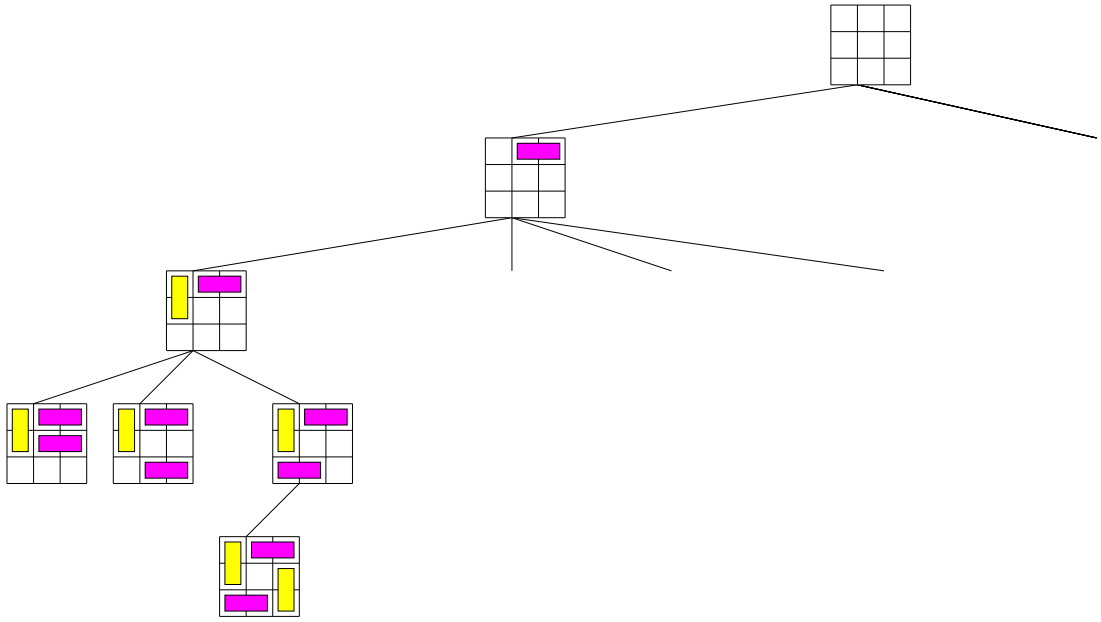
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

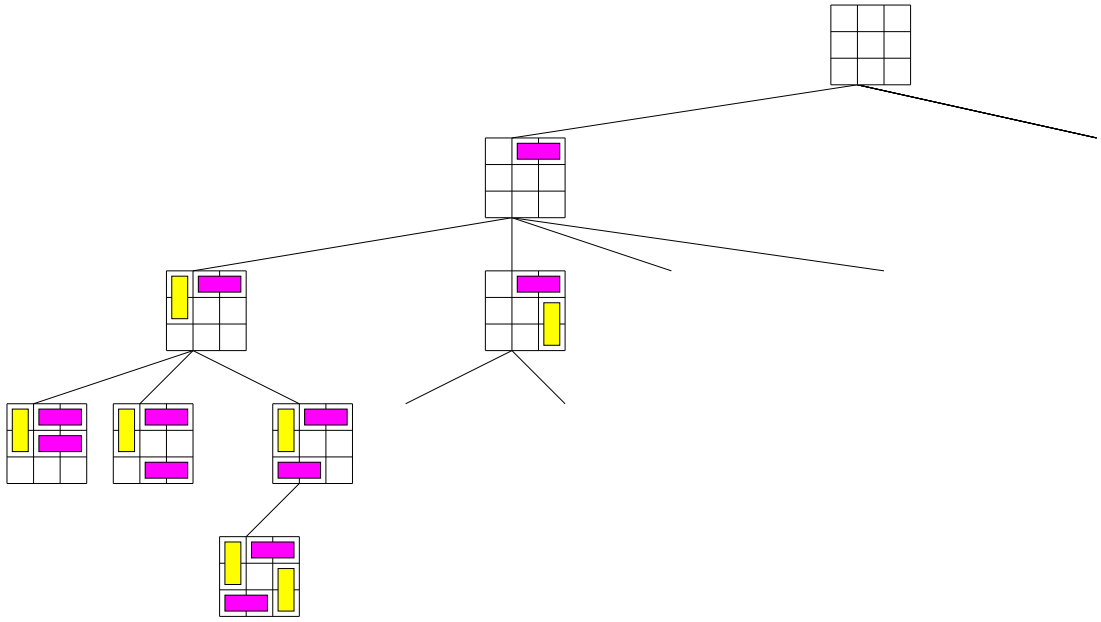
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

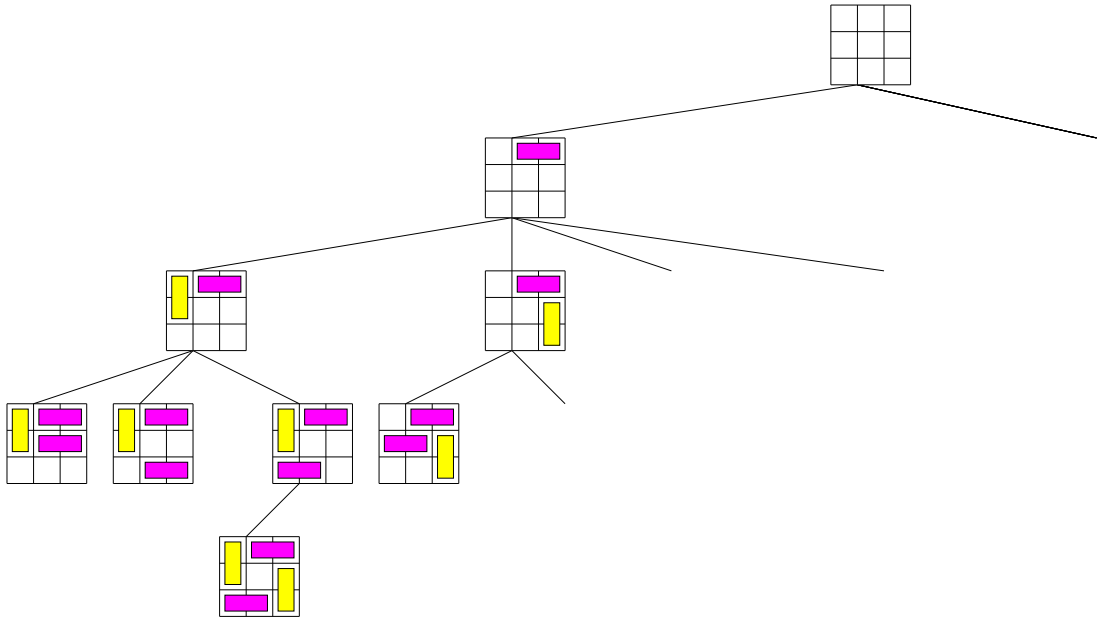
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

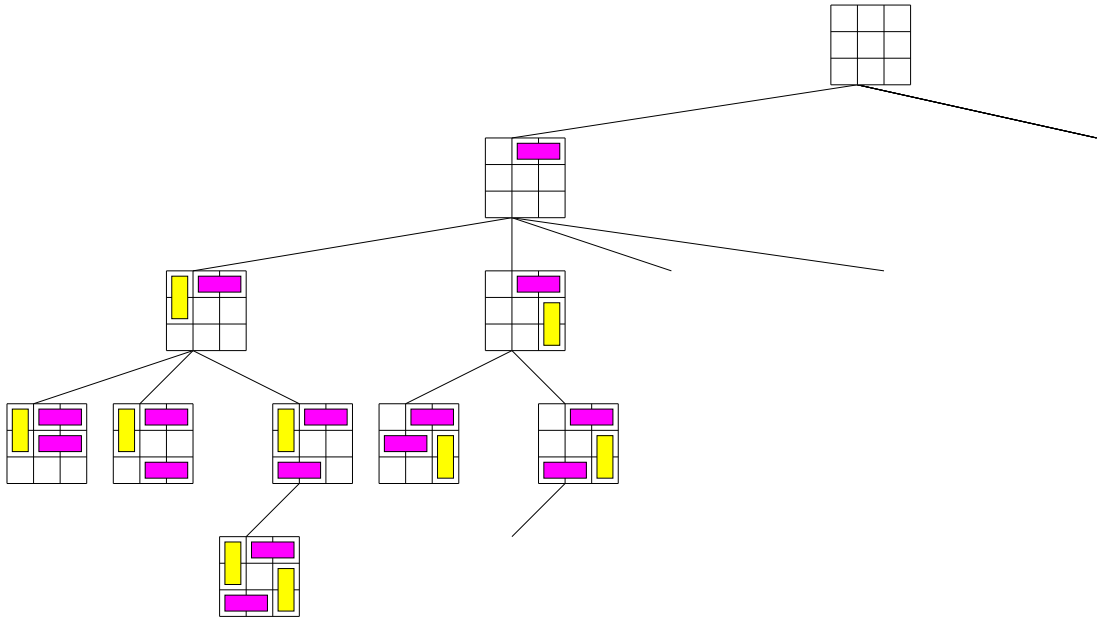
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

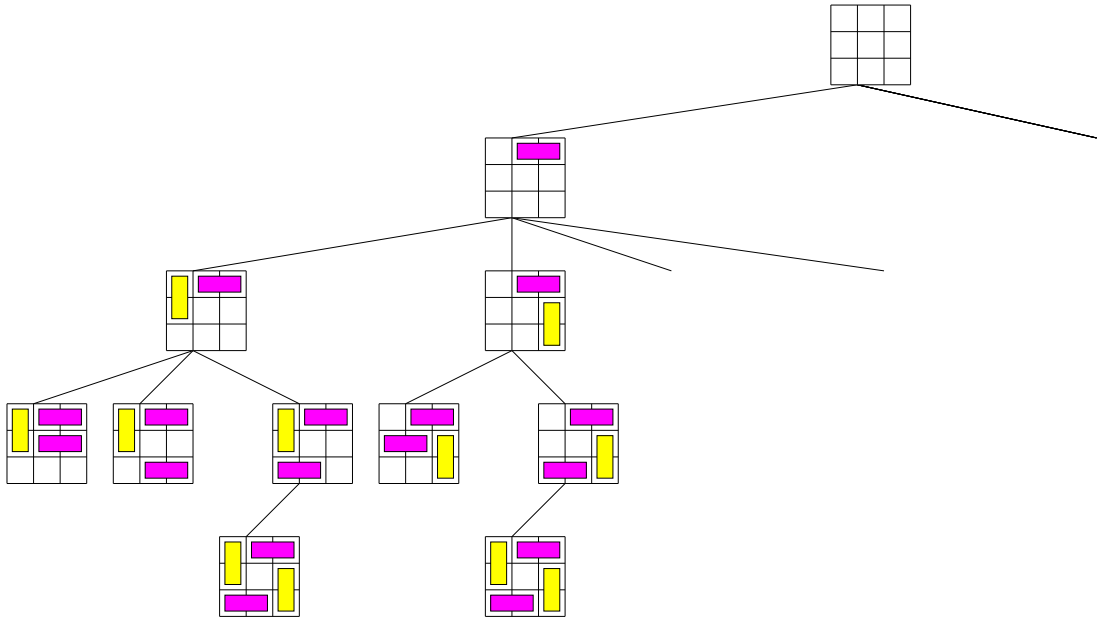
La simulation de tous les coups possibles depuis la position initiale permet de construire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

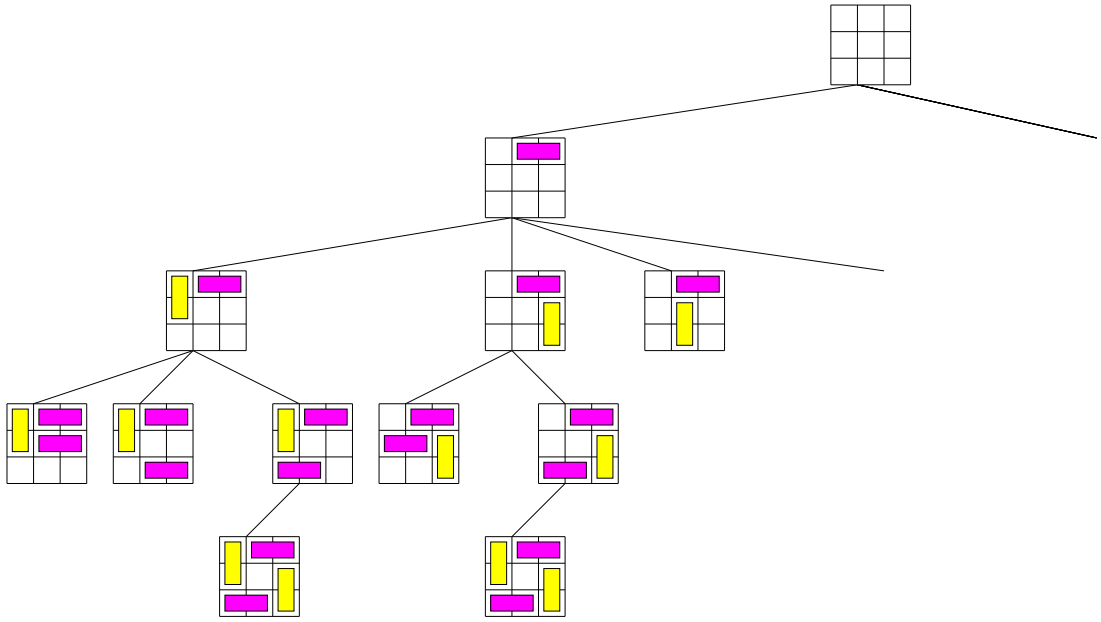
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

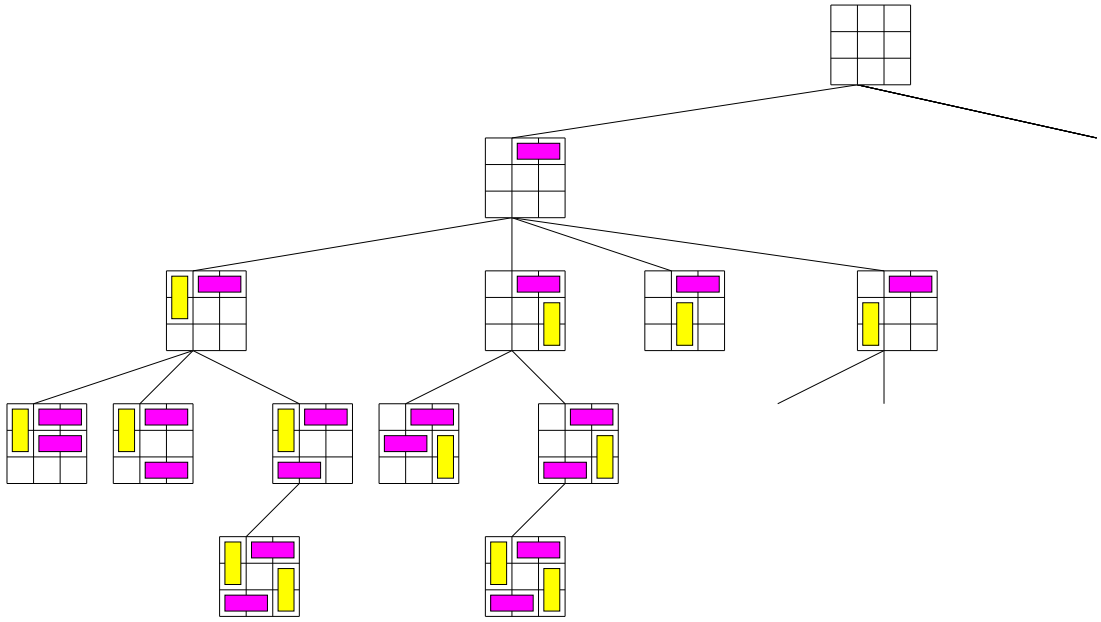
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

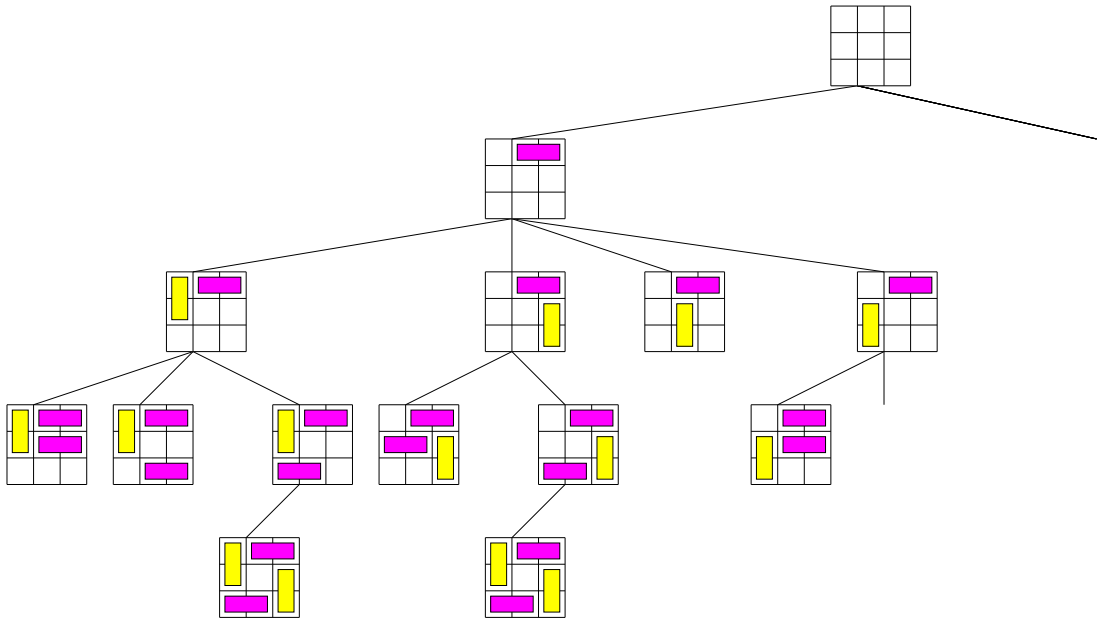
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

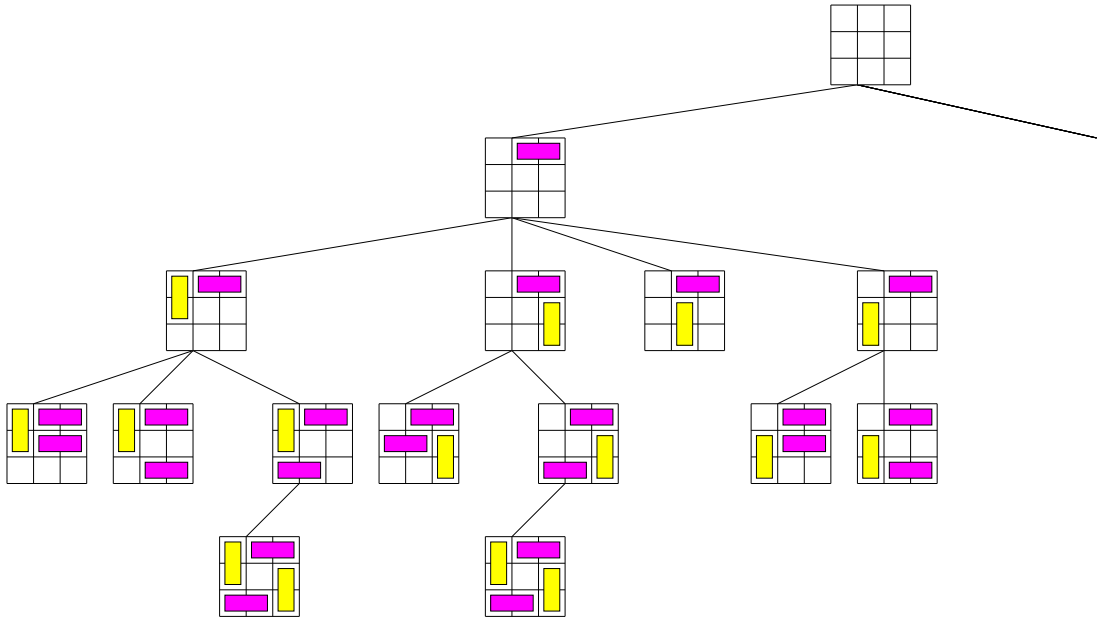
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

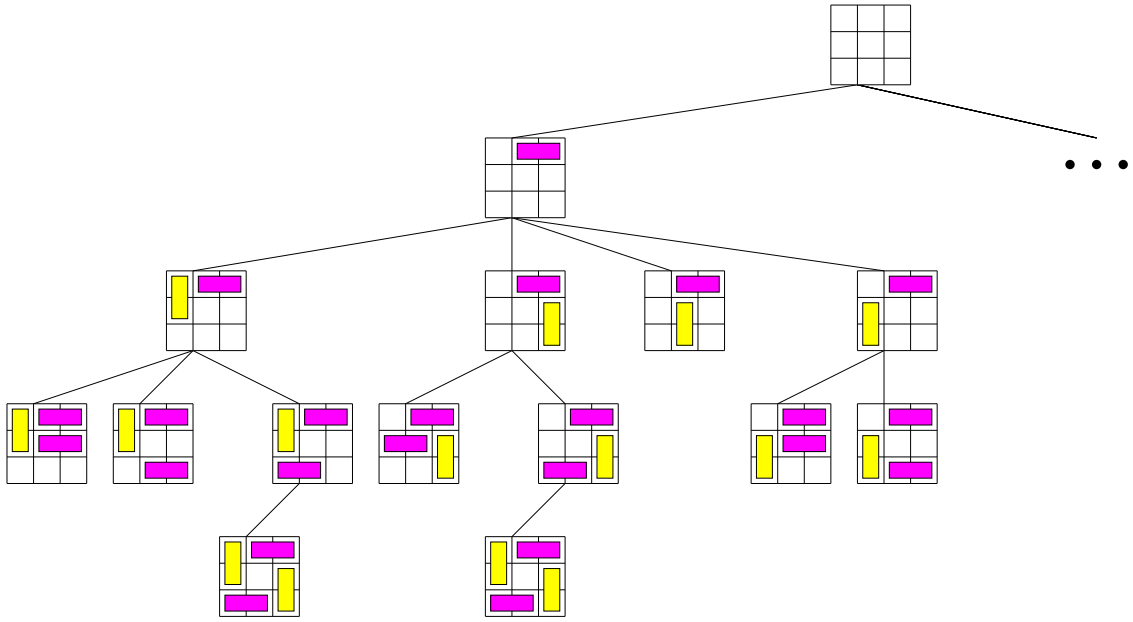
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

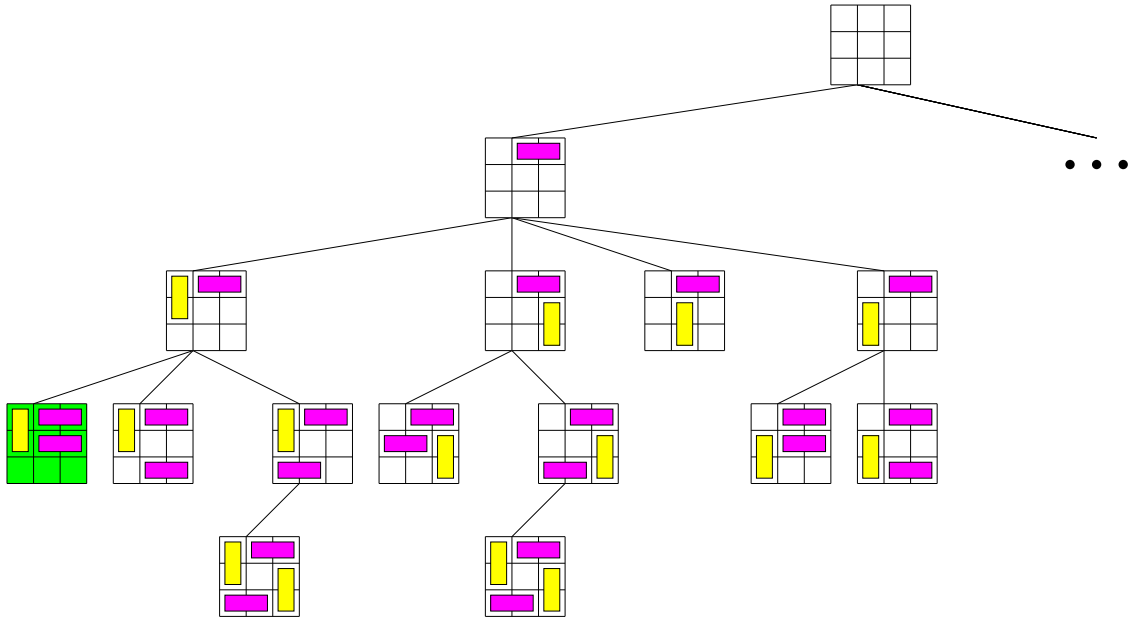
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

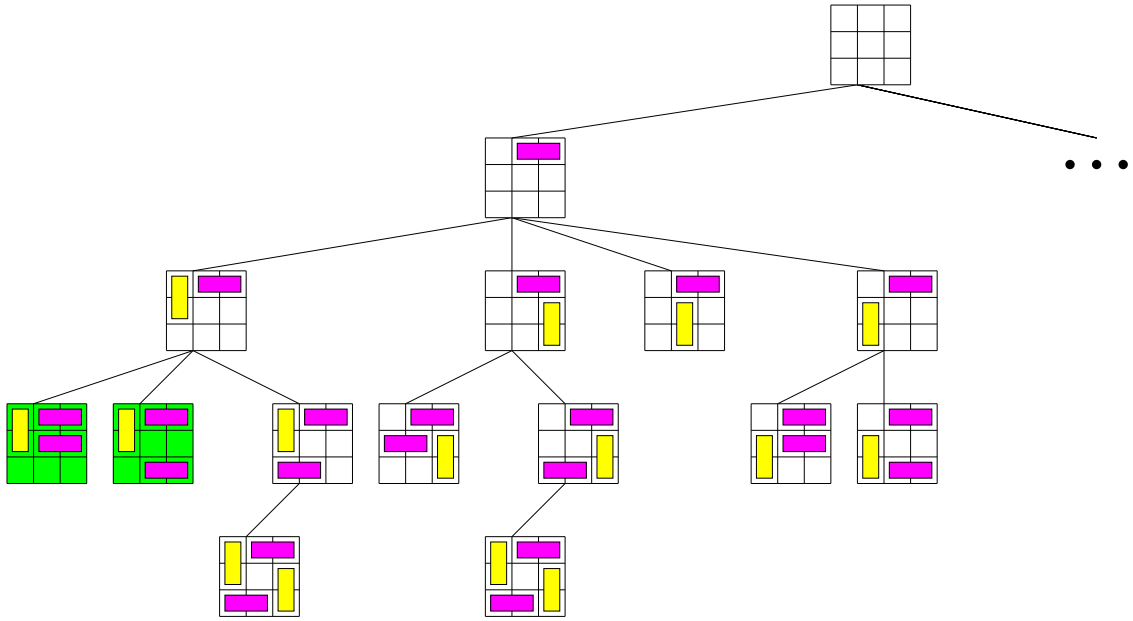
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

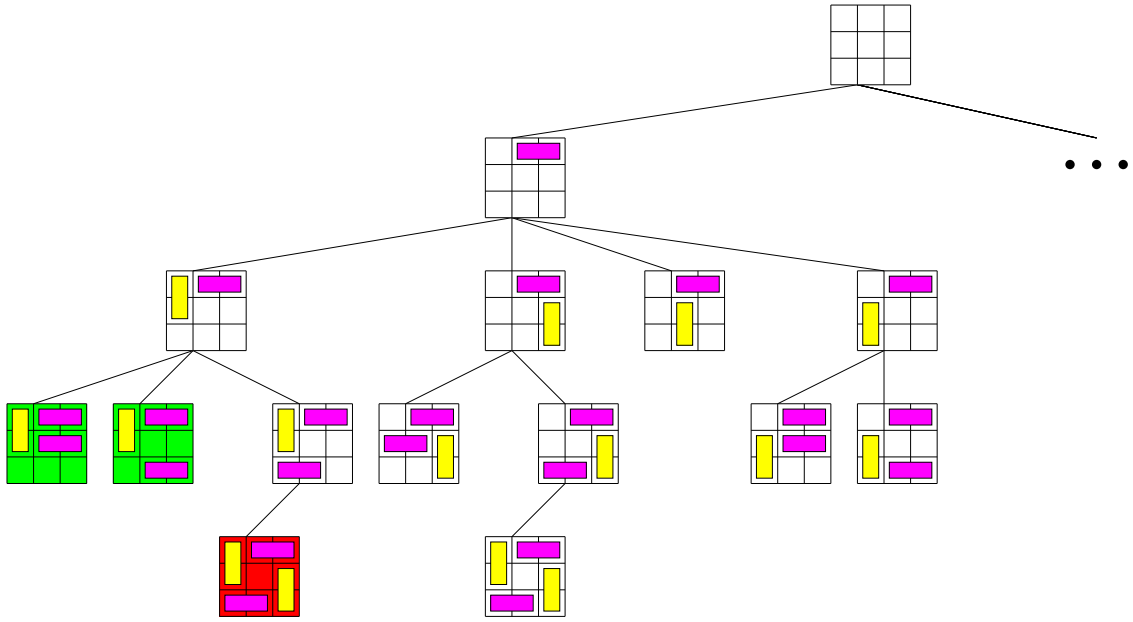
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

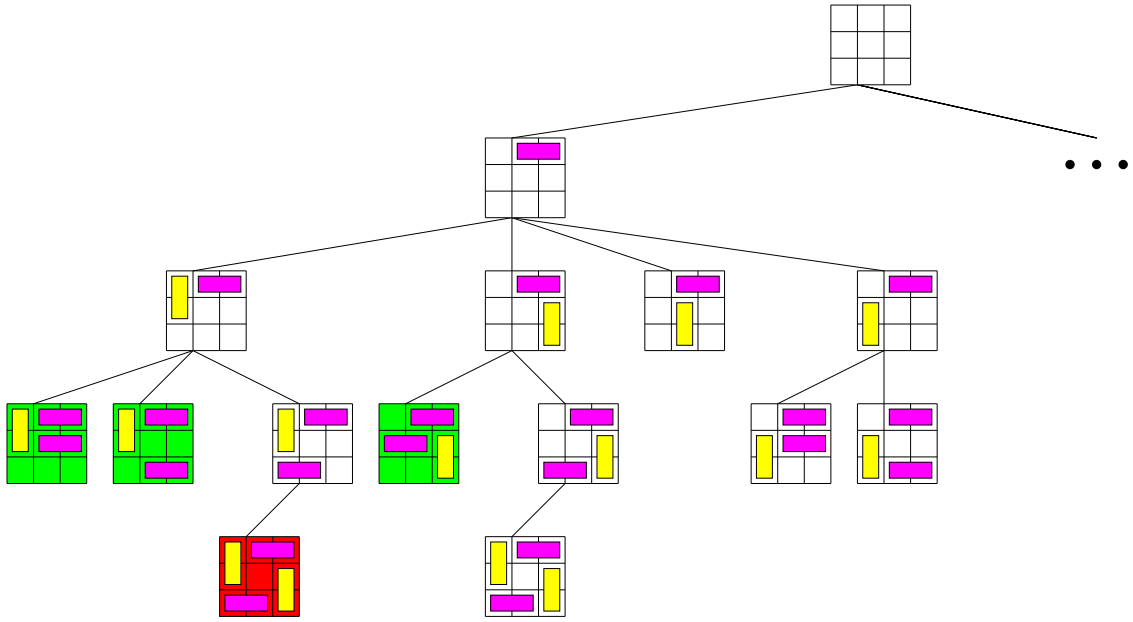
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

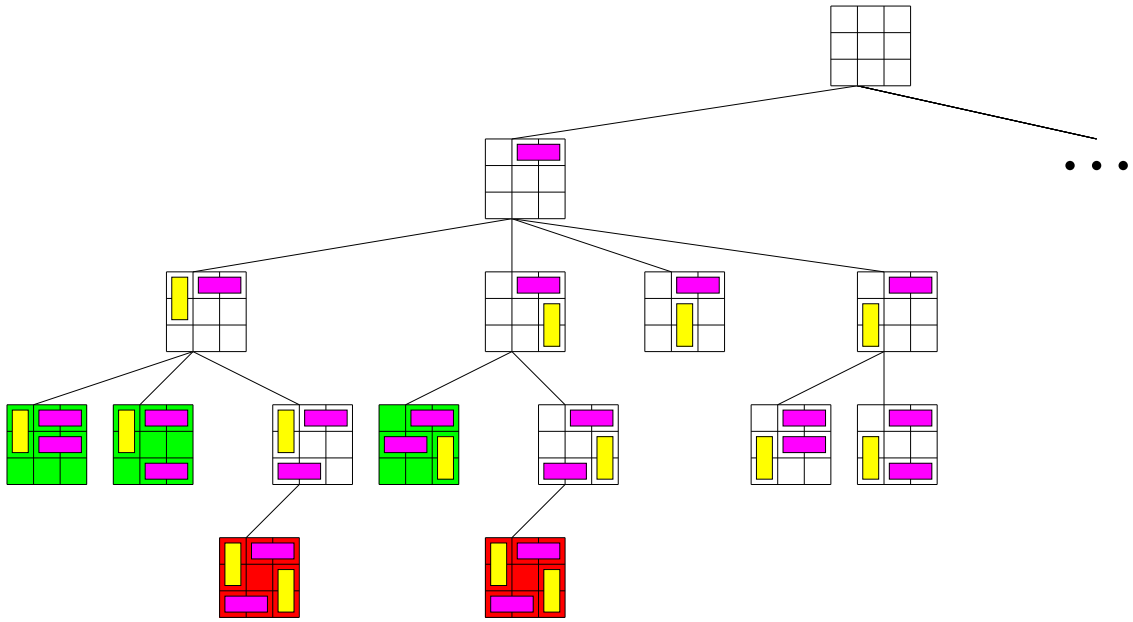
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

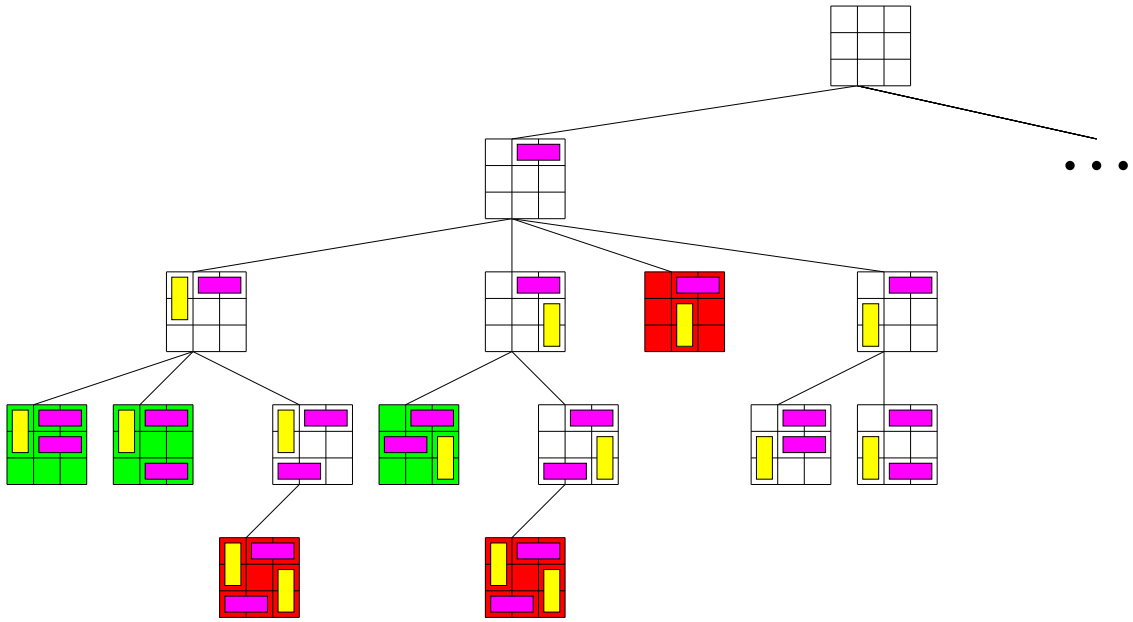
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

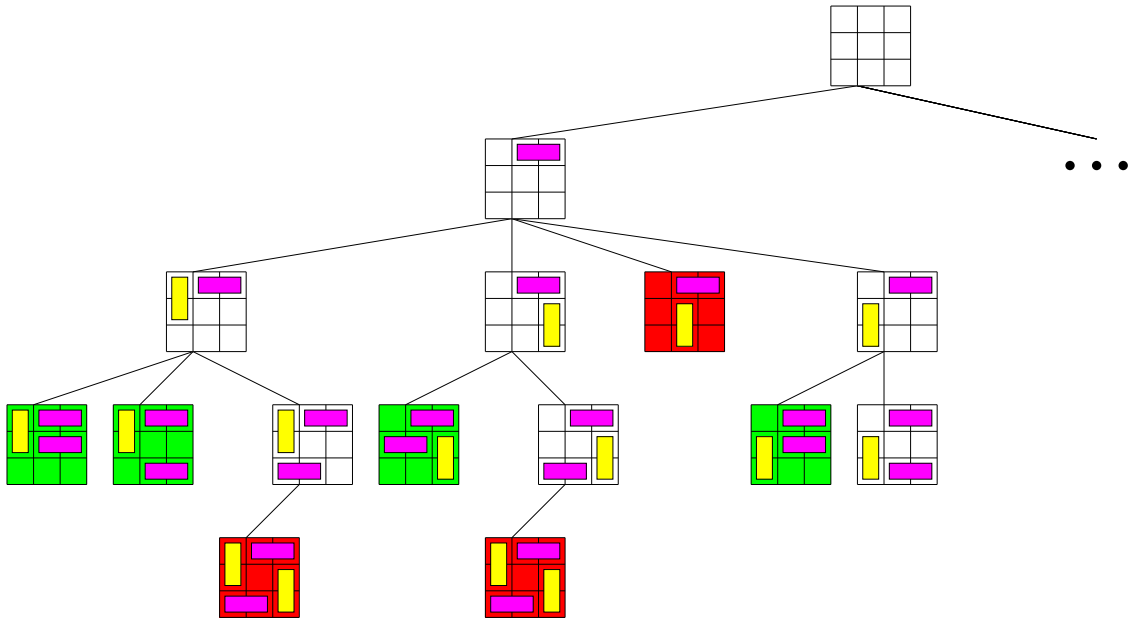
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

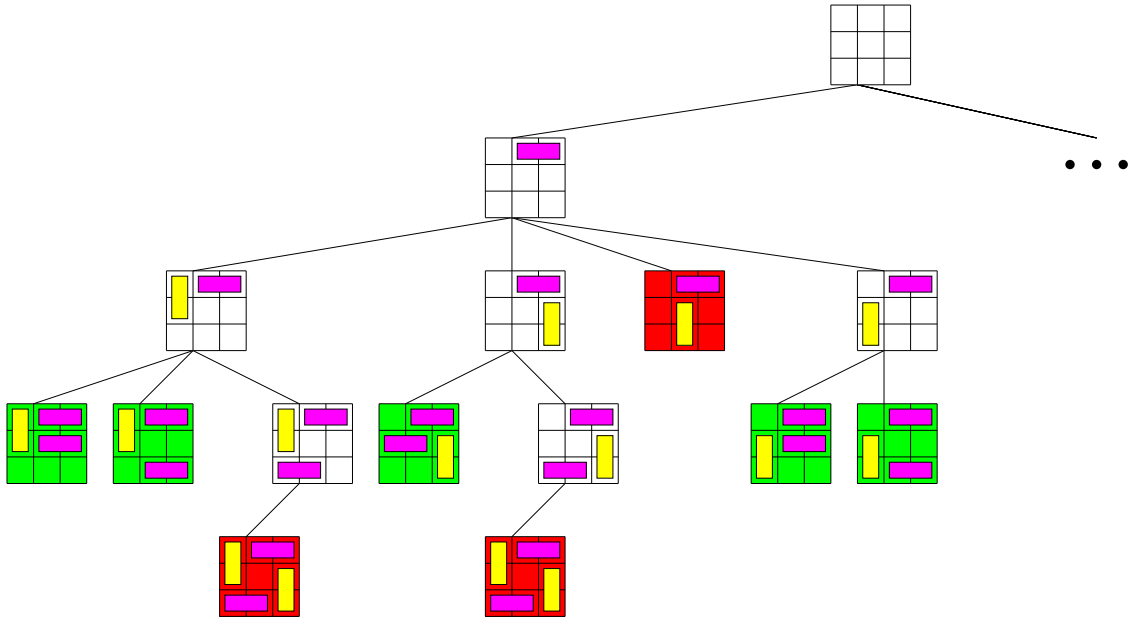
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

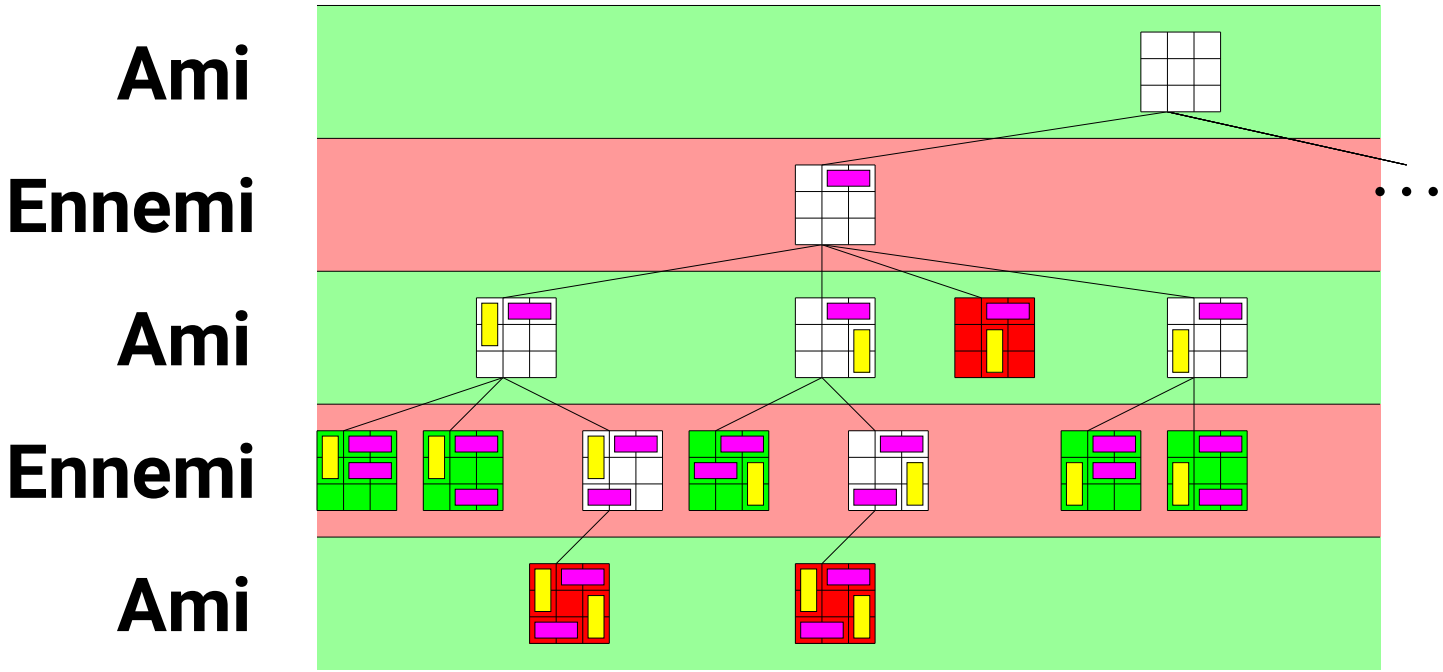
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

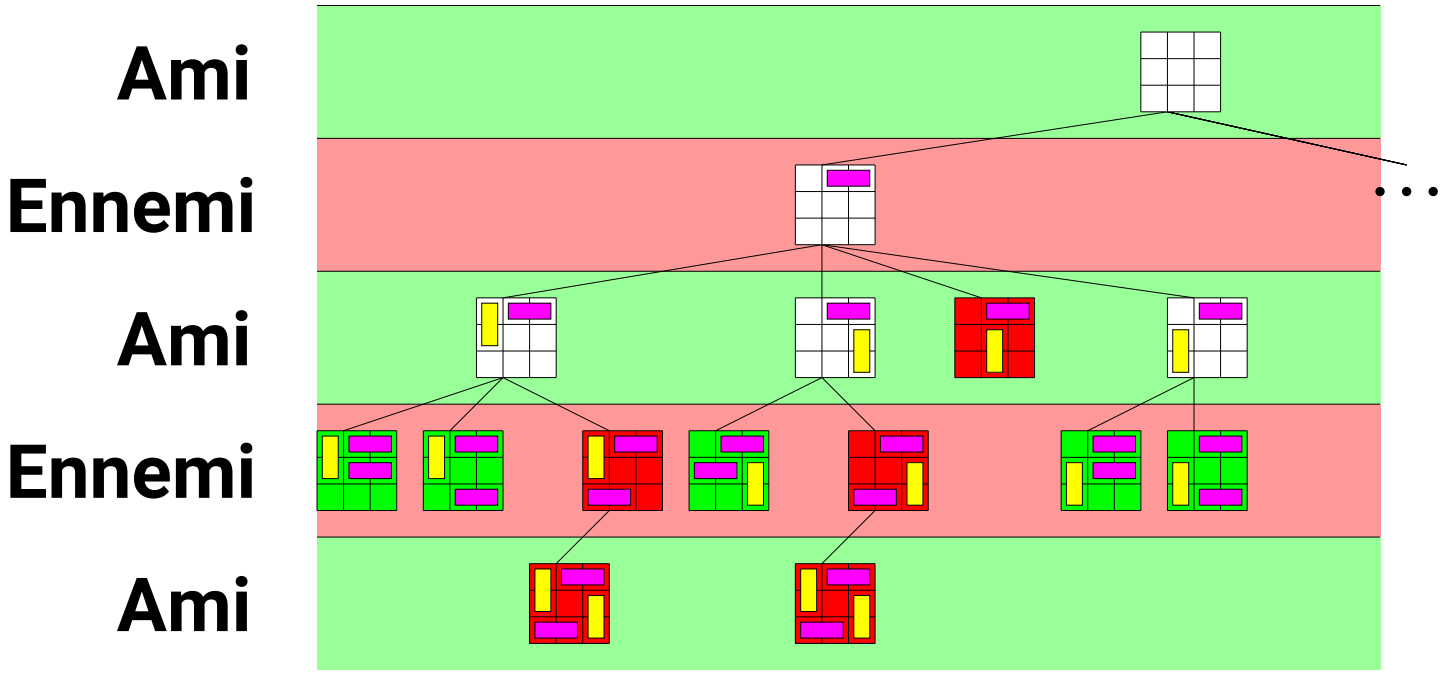
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

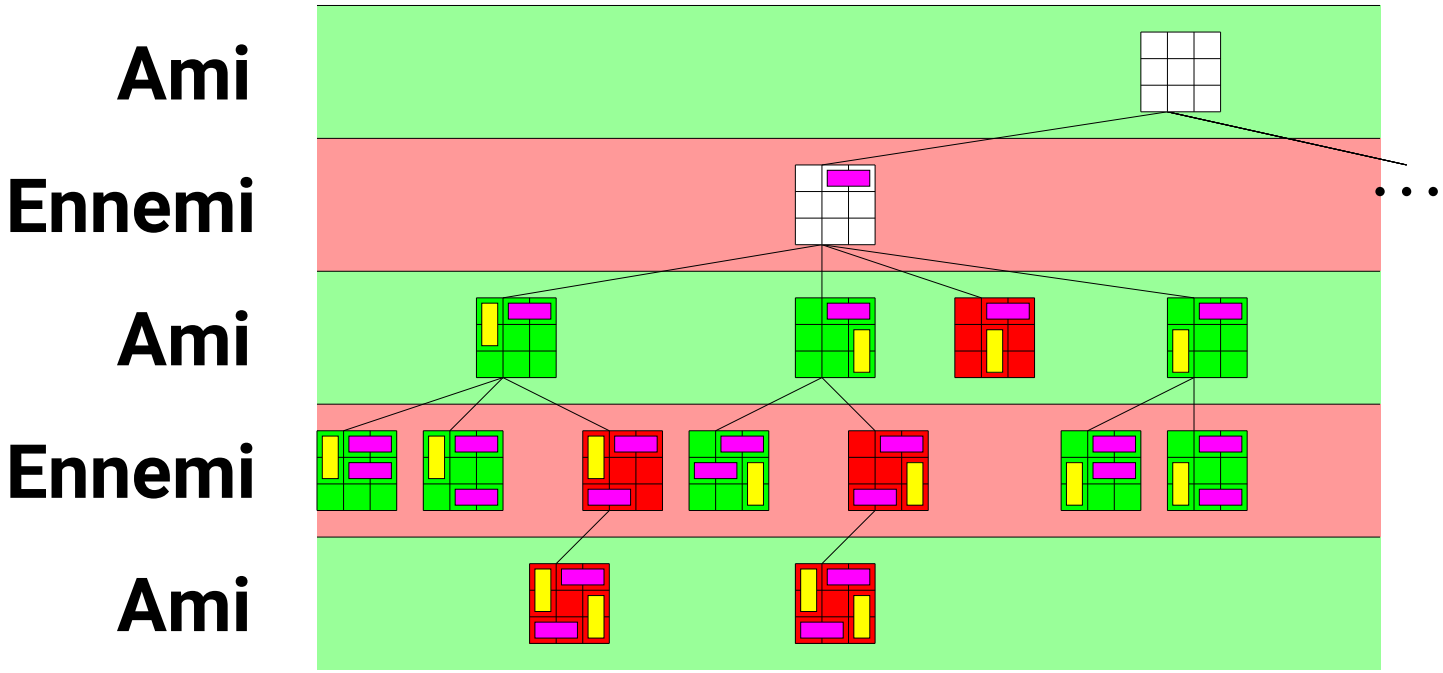
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

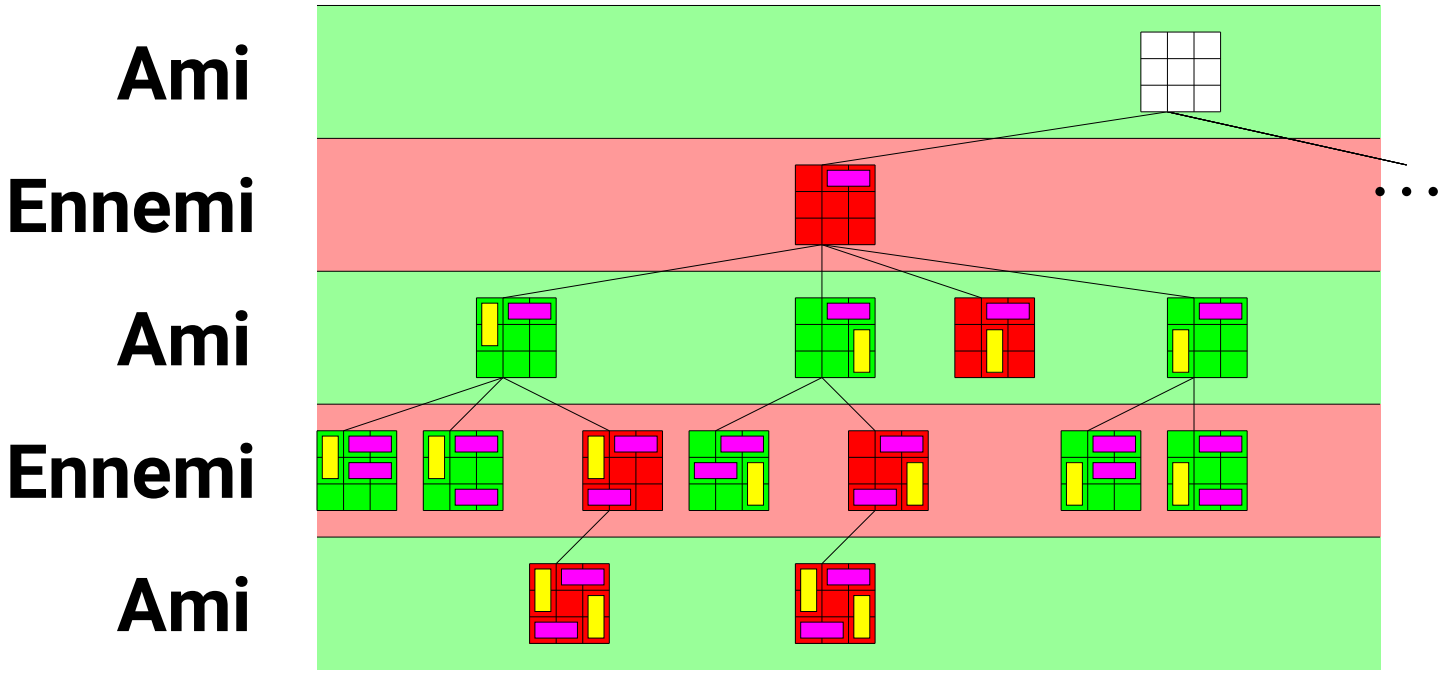
La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

La simulation de tous les coups possibles depuis la position initiale permet de cc :ruire *l'arbre de jeu*.



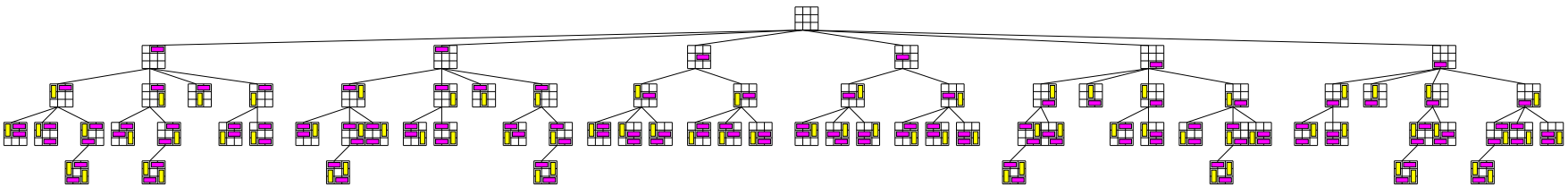
Si Ennemi joue bien, jouer ce coup en premier me fera perdre

Exploration de l'arbre de jeu

Exemple total sur le domino 3 :

Exploration de l'arbre de jeu

Exemple total sur le domino 3 :



Exploration de tout l'arbre de jeu

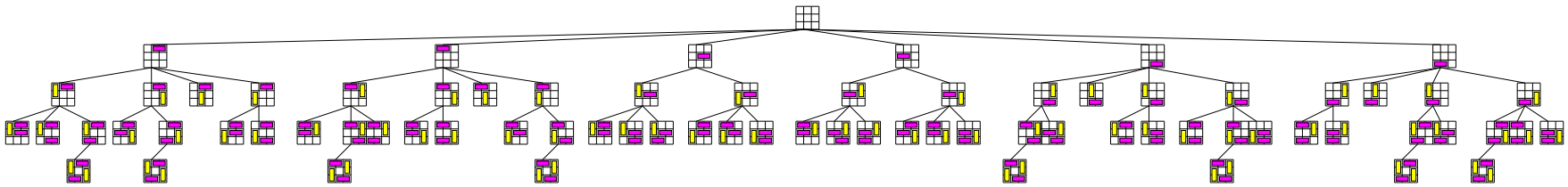
Analyse

L'exploration peut se voir comme un parcours classique en profondeur à main gauche dans un arbre décrit *en intention*.
Dans l'exemple on a trouvé une **Stratégie Gagnante** en 75 noeuds développés.

Question

Jusqu'où pourrait-on aller avec cette méthode sur un problème simple comme celui-ci ?

Domino 4, 5, 6, 10, 50, 500 ?
Des idées ?



Exploration de tout l'arbre de jeu

Analyse

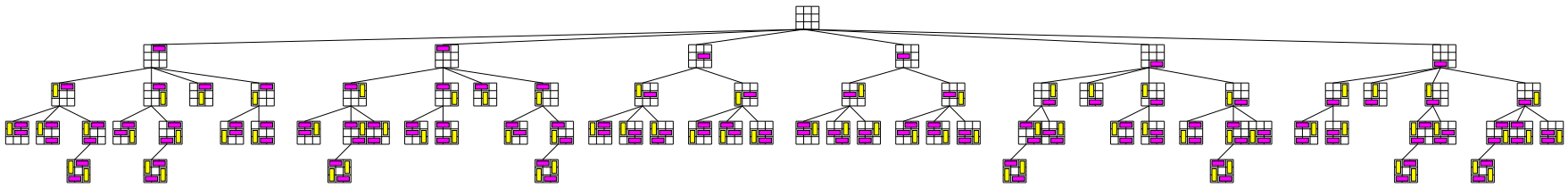
L'exploration peut se voir comme un parcours classique en profondeur à main gauche dans un arbre décrit *en intention*.
Dans l'exemple on a trouvé une **Stratégie Gagnante** en 75 noeuds développés.

Question

Jusqu'où pourrait-on aller avec cette méthode sur un problème simple comme celui-ci ?

Domino 4, 5, 6, 10, 50, 500 ?

Des idées ?



Exploration de tout l'arbre de jeu

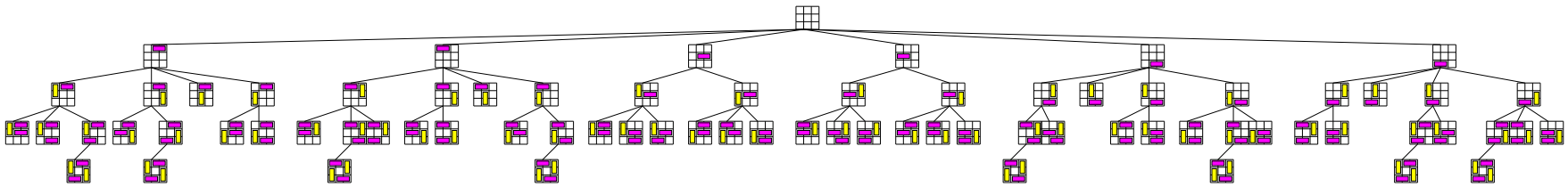
Analyse

L'exploration peut se voir comme un parcours classique en profondeur à main gauche dans un arbre décrit *en intention*.
Dans l'exemple on a trouvé une **Stratégie Gagnante** en 75 noeuds développés.

Question

Jusqu'où pourrait-on aller avec cette méthode sur un problème simple comme celui-ci ?

Domino 4, 5, 6, 10, 50, 500 ?
Des idées ?



Exploration de tout l'arbre de jeu

Impossibilité, même pour des jeux simples

Quelques expérimentations sur un ordinateur réçent, code écrit en C relativement optimisé :

Taille	Nombre de noeuds	Temps
3	75	0.00
4	65 081	0.00
5	2 103 584 600	727.9
6	–	–

On se trouve face à une **véritable explosion combinatoire**

Comment arriver à aller plus loin ?

Exploration de tout l'arbre de jeu

Impossibilité, même pour des jeux simples

Quelques expérimentations sur un ordinateur réçent, code écrit en C relativement optimisé :

Taille	Nombre de noeuds	Temps
3	75	0.00
4	65 081	0.00
5	2 103 584 600	727.9
6	–	–

On se trouve face à une **véritable explosion combinatoire**

Comment arriver à aller plus loin ?

Exploration de tout l'arbre de jeu

Impossibilité, même pour des jeux simples

Quelques expérimentations sur un ordinateur réçent, code écrit en C relativement optimisé :

Taille	Nombre de noeuds	Temps
3	75	0.00
4	65 081	0.00
5	2 103 584 600	727.9
6	–	–

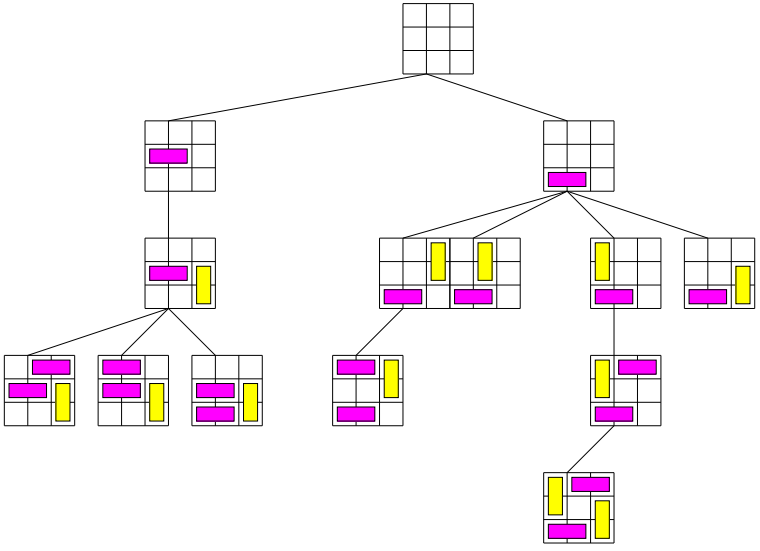
On se trouve face à une **véritable explosion combinatoire**

Comment arriver à aller plus loin ?

Exploration d'un graphe de jeu

Ne pas réexplorer des sous-graphes commun

Idée : dans certains jeux, un même état peut être atteint par différents endroits. Il faut introduire un mécanisme permettant de ne pas réexplorer des sous-arbres déjà vus.
On peut aller plus loin : considérer tous les états symétriques du plateau de jeu.
L'arbre de jeu précédent se réécrit en



Exploration d'un graphe de jeu

Cela ne change pas le fond des choses

Problèmes : il faut introduire un mécanisme pour retrouver les noeuds déjà explorés. Il faut aussi tous les garder en mémoire.
En pratique :

Taille	Arbre	Graphe	Temps (Graphe)
3	75	23	0.00
4	65 081	2 120	0.00
5	2 103 584 600	718 582	83.3
6	–	–	–

En *profilant* l'exécution, tout le temps est maintenant passé à vérifier si un noeud a déjà été vu.

Toujours pas suffisant ! On ne va pas significativement plus loin !

Exploration d'un graphe de jeu

Cela ne change pas le fond des choses

Problèmes : il faut introduire un mécanisme pour retrouver les noeuds déjà explorés. Il faut aussi tous les garder en mémoire.
En pratique :

Taille	Arbre	Graphe	Temps (Graphe)
3	75	23	0.00
4	65 081	2 120	0.00
5	2 103 584 600	718 582	83.3
6	–	–	–

En *profilant* l'exécution, tout le temps est maintenant passé à vérifier si un noeud a déjà été vu.

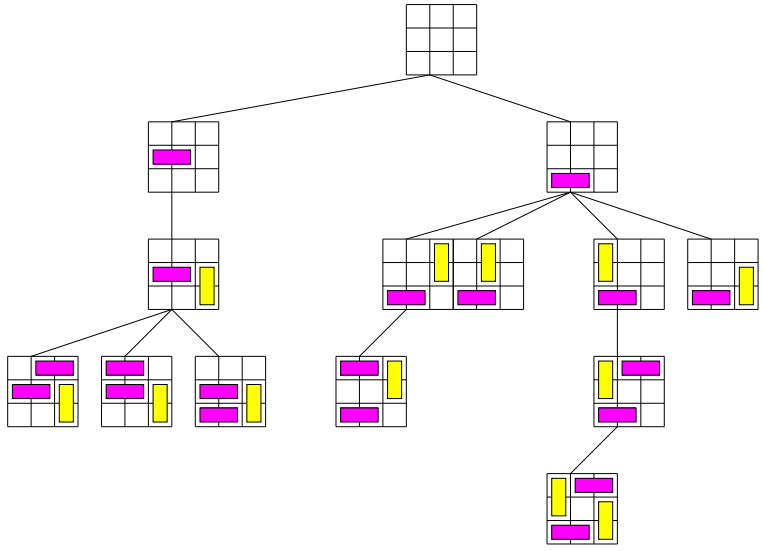
Toujours pas suffisant ! On ne va pas significativement plus loin !

Une première recherche *intelligente*

Les premières *coupes* pas trop idiotes

Idée

Si une branche *amie* mène à la victoire à coups sûrs, ne pas développer les branches voisines.
Si une branche *ennemie* mène à la défaite, ne pas développer les branches voisine.



Une première recherche *intelligente*

Les premières *coupes* pas trop idiotes

- Trouve s'il existe *une stratégie gagnante*
- Permet de couper (élaguer) des parties entières de l'arbre
- Garde les propriétés de l'exploration complète de l'arbre

- Oblige à aller jusqu'aux feuilles de l'arbre (déroulement de toute une partie sur chaque branche non coupée)
- En pratique, aucun jeux ne permet cette technique (ou alors en toute fin de partie)

Toujours pas suffisant ! Il faut pouvoir couper l'arbre sans devoir aller jusqu'aux feuilles

recherche avec horizon

Introduction d'*heuristiques*

Idée

On ne développe l'arbre de recherche que jusqu'à une certaine profondeur maximale p. Les feuilles de l'arbre ne sont plus forçément des positions finales du jeu.
On ne peut donc plus se contenter de l'information « gagné » ou « perdu ». Il faut **évaluer** les positions.

Définition intuitive

Les heuristiques sont des fonctions statiques associant un réel aux plateaux de jeu. Dans les jeux avec adversaire, plus l'heuristique est grande et positive, plus on est proche de la victoire. Plus elle est négative et petite, plus on est proche de la défaite.

Exemple des dominos 3×3

Idée

- On développe l'arbre (ou graphe) de recherche jusqu'à une profondeur donnée (déterminée suivant le temps que l'on a pour jouer et le facteur de branchement estimé du jeu)
- On évalue la position aux feuilles (dépend du plateau et du joueur)
- On fait remonter l'évaluation pour *trouver le meilleur coup garantissant au moins l'estimation de la feuille*

Exemple des dominos 3×3

Idée

- On développe l'arbre (ou graphe) de recherche jusqu'à une profondeur donnée (déterminée suivant le temps que l'on a pour jouer et le facteur de branchement estimé du jeu)
- On évalue la position aux feuilles (dépend du plateau et du joueur)
- On fait remonter l'évaluation pour *trouver le meilleur coup garantissant au moins l'estimation de la feuille*

Exemple des dominos 3×3

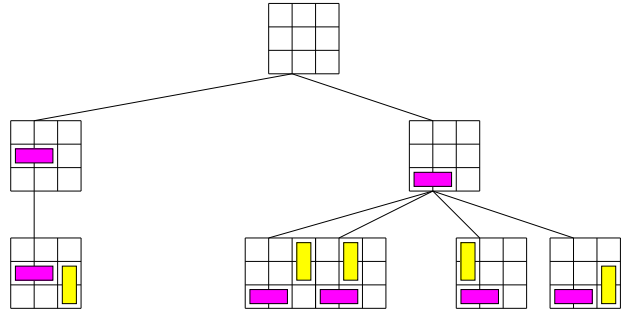
Idée

- On développe l'arbre (ou graphe) de recherche jusqu'à une profondeur donnée (déterminée suivant le temps que l'on a pour jouer et le facteur de branchement estimé du jeu)
- On évalue la position aux feuilles (dépend du plateau et du joueur)
- On fait remonter l'évaluation pour *trouver le meilleur coup garantissant au moins l'estimation de la feuille*

Exemple des dominos 3×3

Idée

- On développe l'arbre (ou graphe) de recherche jusqu'à une profondeur donnée (déterminée suivant le temps que l'on a pour jouer et le facteur de branchement estimé du jeu)
- On évalue la position aux feuilles (dépend du plateau et du joueur)
- On fait remonter l'évaluation pour *trouver le meilleur coup garantissant au moins l'estimation de la feuille*



Exemple d'heuristiques : les Échecs dès 1950

(1)The relative values of queen, rook, bishop, knight and pawn are about 9, 5, 3, 3, 1, respectively. Thus other things being equal (!) if we add the numbers of pieces for the two sides with these coefficients, the side with the largest total has the better position.
(2)Rooks should be placed on open files. This is part of a more general principle that the side with the greater mobility, other things equal, has the better game.
(3)Backward, isolated and doubled pawns are weak.
(4)An exposed king is a weakness (until the end game).

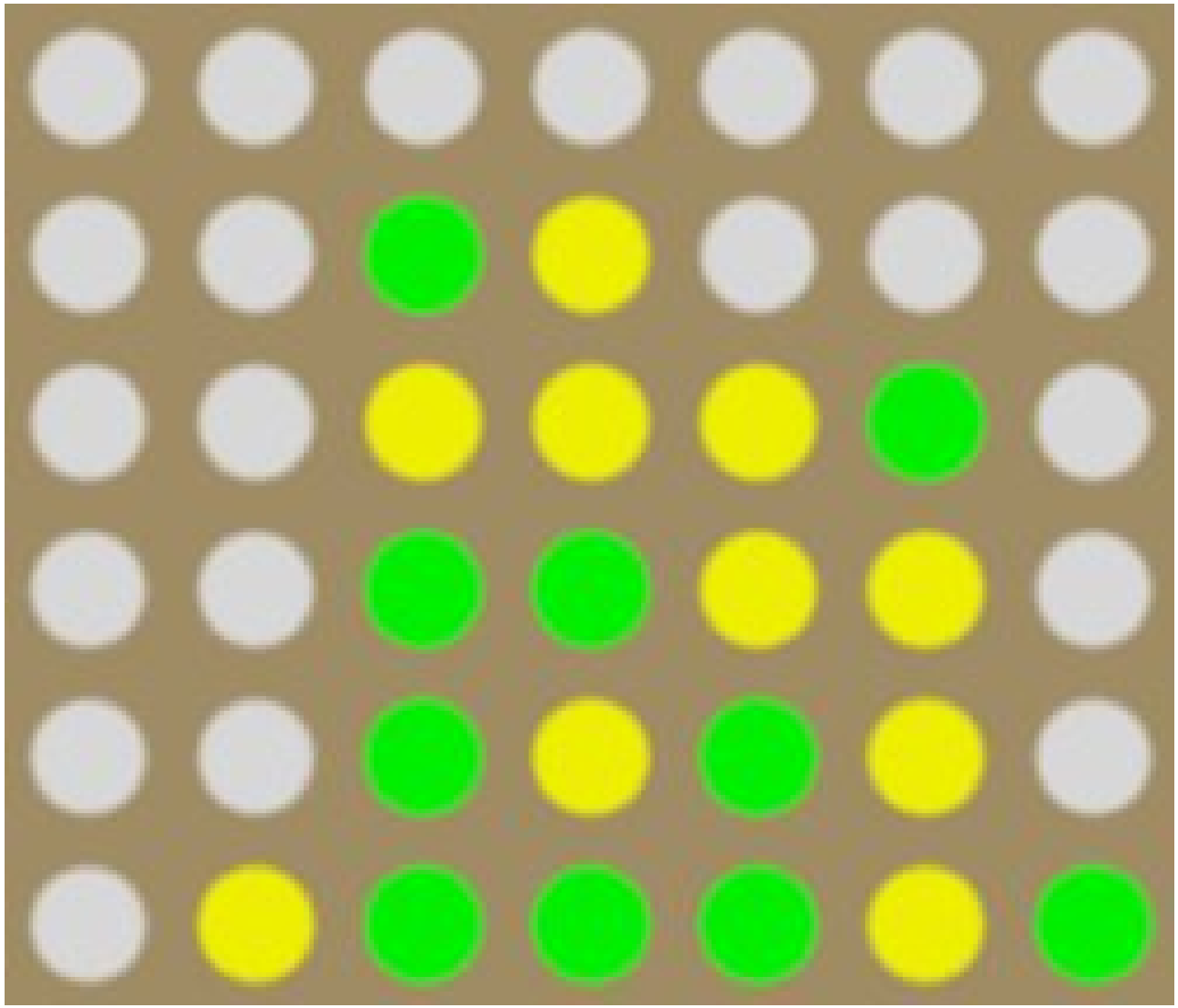
These and similar principles are only generalizations from empirical evidence of numerous games, and only have a kind of statistical validity. Probably any chess principle can be contradicted by particular counter examples. However, form these principles one can construct a crude evaluation function. The following is an example: -

$$f(P) = 200(K-K') + 9(Q-Q') + 5(R-R') + 3(B-B'+N-N') + (P-P') - 0.5(D-D'+S-S'+I-I') + 0.1(M-M') + \ldots$$

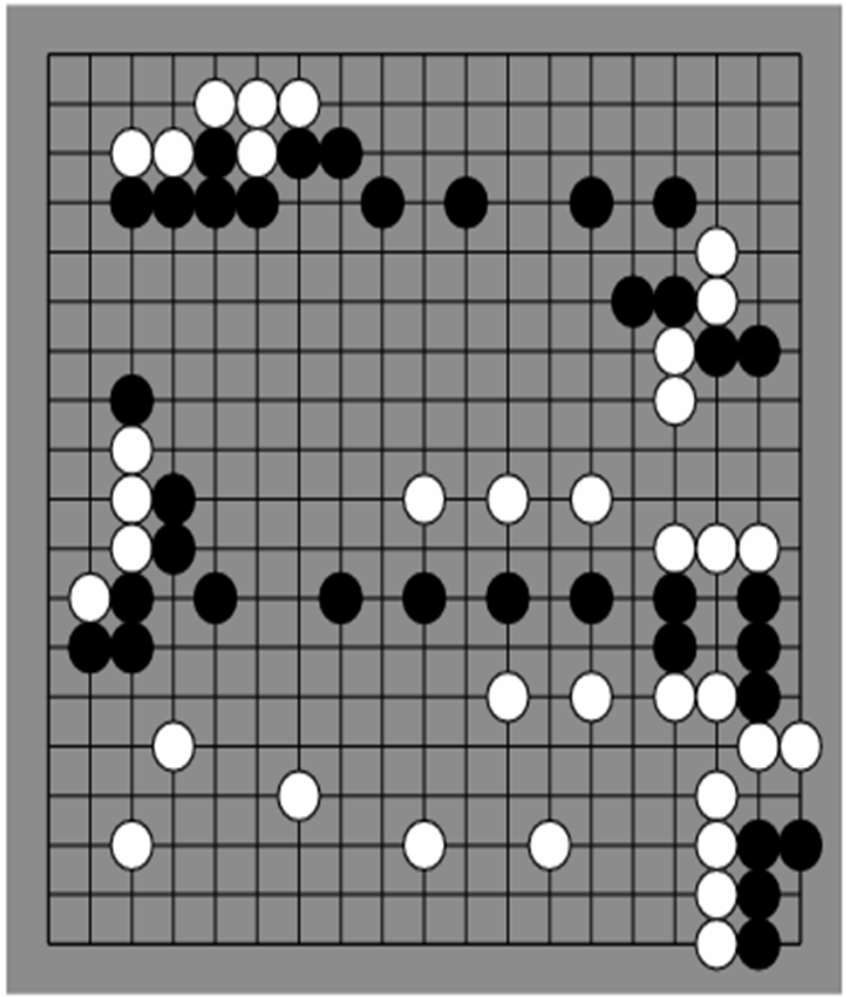
in which: -

(1)K,Q,R,B,B,P are the number of White kings, queens, rooks, bishops, knights and pawns on the board.
(2)D,S,I are doubled, backward and isolated White pawns.
(3)M= White mobility (measured, say, as the number of legal moves available to White).

Exemple d'heuristiques : Puissance 4



Exemple d'heuristiques : Go



Introduction d'*heuristiques*

Nouveau but

Trouver la branche permettant de maximiser la valeur heuristique obtenue sur le plateau après les p prochains coups.

Supposition forte

On suppose que les deux joueurs jouent « bien » s'ils suivent les indications de l'heuristique.

En pratique, il faudra prendre en compte cela pour des raisons de performances. À chaque feuille de l'arbre, on devra calculer sa valeur heuristique

Problème : Comment faire remonter la meilleur feuille à la racine ?

Introduction d'*heuristiques*

Nouveau but

Trouver la branche permettant de maximiser la valeur heuristique obtenue sur le plateau après les p prochains coups.

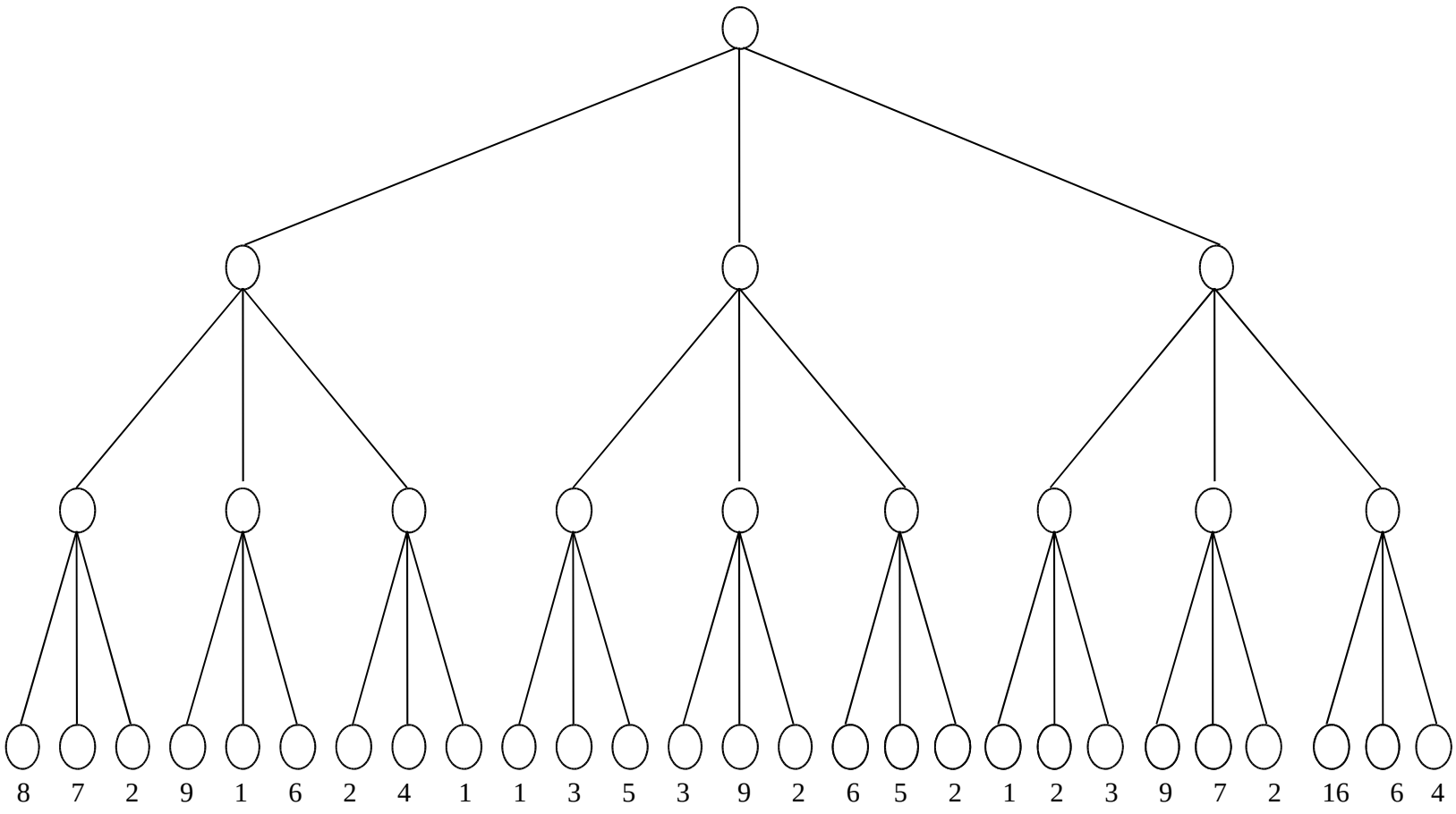
Supposition forte

On suppose que les deux joueurs jouent « bien » s'ils suivent les indications de l'heuristique.

En pratique, il faudra prendre en compte cela pour des raisons de performances. À chaque feuille de l'arbre, on devra calculer sa valeur heuristique

Problème : Comment faire remonter la meilleur feuille à la racine ?

Exemple d'arbre de jeu



Algorithme MiniMax – Fonction $MaxMin$

Idée

On appelle $MaxMin$ pour évaluer la valeur MiniMax de la racine. On doit aussi remonter*le meilleur coups à jouer* qui lui est associé.

```
1 : Fonction MaxMin(etat)                          ▷ Évaluation niveau AMI
2 :     etat : Plateau de jeu courant

3 :     Si EstFeuille(etat) Alors
4 :         Retourner evalue(AMI, etat)            ▷ Évaluation heuristique
5 :     Fin Si
6 :     Meilleur ← −∞
7 :     Pour Tout successeur s de etat Faire
8 :         Meilleur ← max(Meilleur,MinMax(s))
9 :     Fin Pour
10 :    Retourner Meilleur
11 : Fin Fonction
```

Algorithme MiniMax – Fonction $MinMax$

Attention

Si l'évaluation de la position a lieu sur un niveau impair (c'est à $ENNEMI$ de jouer), la fonction heuristique doit en tenir compte !

```
1 : Fonction MinMax(etat)                    ▷ Évaluation niveau ENNEMI
2 :     etat : Plateau de jeu courant

3 :     Si EstFeuille(etat) Alors
4 :         Retourner evalue(ENNEMI, etat)   ▷ Évaluation heuristique
5 :     Fin Si
6 :     Pire ← +∞
7 :     Pour Tout successeur s de etat Faire
8 :         Pire ← min(Pire,MaxMin(s))
9 :     Fin Pour
10 :    Retourner Pire
```

Les deux fonctions MiniMax

Fonction $MaxMin(etat)$ ▷ Évaluation niveau AMI
  $etat$: Plateau de jeu courant

  Si $EstFeuille(etat)$ Alors
    Retourner $evalue(AMI, etat)$ ▷ Évaluation heuristique
  Fin Si
  $Meilleur \leftarrow -\infty$
  Pour Tout successeur s de $etat$ Faire
    $Meilleur \leftarrow max(Meilleur, MinMax(s))$
  Fin Pour
  Retourner $Meilleur$
Fin Fonction

Fonction $MinMax(etat)$ ▷ Évaluation niveau ENNEMI
  $etat$: Plateau de jeu courant

  Si $EstFeuille(etat)$ Alors
    Retourner $evalue(ENNEMI, etat)$ ▷ Évaluation heuristique
  Fin Si
  $Pire \leftarrow +\infty$
  Pour Tout successeur s de $etat$ Faire
    $Pire \leftarrow min(Pire, MaxMin(s))$
  Fin Pour
  Retourner $Pire$

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :

  Dilemme du prisonnier

	Silence	Accusation
Silence	0, 0	10, 1
Accusation	1, 10	5, 5

- Amplification de valeurs heuristiques

  Une valeur faible mais isolée dans une zone tranquille sera préférée à une valeur presque identique dans une zone plus « animée ».

  En pratique cet effet est contre-balancé par les valeurs heuristiques qui font désespérément diverger les feuilles.
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :

  Dilemme du prisonnier

	Silence	Accusation
Silence	0, 0	10, 1
Accusation	1, 10	5, 5

- Amplification de valeurs heuristiques

  Une valeur faible mais isolée dans une zone inaccessible sera préférée à une valeur presque identique dans une zone plus « facile ».

  En pratique cet effet est contre-balancé par les valeurs heuristiques qui sont très fortement éloignées des feuilles.
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone inhospitalière sera préférée à une valeur presque identique dans une zone plus viable.
  - En pratique cet effet est contre-balancé par les incertitudes heuristiques qui font généralement éloigner les feuilles.
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

  - Amplification de valeurs heuristiques

    Une valeur faible mais isolée dans une zone d'ambiguïté sera préférée à une valeur presque identique dans une zone plus « calme ».

    En pratique cet effet est contre-balancé par les valeurs heuristiques qui font très rapidement diverger les résultats.
  - Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone dangereuse sera préferée à une valeur presque identique dans une zone plus « amie »
  - En pratique cet effet est compensé par les heuristiques qui font disparaître rapidement les mauvais coups
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone dangereuse sera préférée à une valeur presque identique dans une zone plus « amie »
  - En pratique cet effet est contre-balancé par les valeurs heuristiques qui sont théoriquement éloignées aux feuilles.
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone dangereuse sera préferée à une valeur presque identique dans une zone plus « amie »
  - En pratique cet effet est contre-balancé par les valeurs heuristiques qui sont théoriquement éloignées aux feuilles.
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone dangereuse sera préferée à une valeur presque identique dans une zone plus « amie »
  - En pratique cet effet est contre-balancé par les valeurs heuristiques qui sont théoriquement éloignées aux feuilles.
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Limitations des principes MiniMax

- Que se passe-t-il si un joueur
  - Ne joue pas « bien » ?
  - Ne joue pas selon l'estimation heuristique ?
- Problèmes de collaboration :
  - Dilemme du prisonnier

	Silence	Accusation
Silence	0 0	10 1
Accusation	1 10	5 5

- Amplification de valeurs heuristiques
  - Une valeur faible mais isolée dans une zone dangereuse sera préferée à une valeur presque identique dans une zone plus « amie »
  - En pratique cet effet est contre-balancé par les valeurs heuristiques qui sont théoriquement éloignées aux feuilles.
- Le résultat est le même, après n'importe quelle application d'une fonction monotone sur les valeurs heuristiques.

Recherche dans les arbres de jeux

L'espace des états atteignables peut être gigantesque. Il dépend :

- Facteur de branchement b du jeu
- Nombre de coups n d'une partie

Idée des tailles de certains jeux

- Échecs : p ~ 35 et n ~ 30 coups au minimum
  |graphe d'états| = 35^{30}
  Soit 2099139642966190174953146230280399322509765625
- Othello : $5 \leq b \leq 15$
- Go : b ~ 360

Il faut retrouver l'idée d'élagage de l'arbre de recherche.

$\alpha\beta$
Elagage efficace de l'arbre

Élagage admissible

On veut trouver la même valeur d'évaluation finale du noeud racine sans développer tout l'arbre. Il faut donc élaguer des parties de l'arbre de recherche qui sont sans conséquence sur l'évaluation d'un noeud.

Intuitivement

Soit un noeud n dans l'arbre de recherche, tel que *Joueur* peut joueur en n.
S'il existe pour *Joueur* un choix m meilleur que n (soit à partir du noeud parent de n, soit plus haut dans l'arbre), n ne sera jamais effectivement joué.

$\alpha\beta$
Elagage efficace de l'arbre II

Deux types de coupes : α et β

- α : le meilleur choix à un instant donné pour Max sur le chemin développé. **La valeur α est croissante**
- β : le meilleur choix à un instant donné pour Min sur le chemin développé. **La valeur β est décroissante**

Les coupes auront lieu dès que α est supérieur à β

Algorithme $\alpha\beta$

La première fonction : *MaxValue*

```
1 : Fonction MaxValue(etat, α, β))                    ▷ Évaluation niveau AMI
2 :     etat : Plateau de jeu courant
3 :     α : Meilleur évaluation courante pour AMI
4 :     β : Meilleur évaluation courante pour ENNEMI

5 :     Si EstFeuille(etat) Alors
6 :         Retourner evalue(etat)                    ▷ Évaluation heuristique
7 :     Fin Si
8 :     Pour Tout successeur s de etat Faire
9 :         α ← max(α,MinValue(s, α, β))
10 :        Si α ≥ β Alors                            ▷ Coupe β
11 :            Retourner β
12 :        Fin Si
13 :    Fin Pour
14 :    Retourner α
15 : Fin Fonction
```

Algorithme $\alpha\beta$

La suite : *MinValue*

À bien noter

- Pour évaluer un plateau, on appelle *MaxValue* avec : plateau à évaluer, $\alpha = -\infty$ et $\beta = +\infty$
- Les variables α et β sont bien *locales*
- Elles sont changées par l'intermédiaire des valeurs de retour

```
1 : Fonction MinValue(etat, α, β))                    ▷ Évaluation niveau ENNEMI
2 :     etat : Plateau de jeu courant
3 :     α : Meilleur évaluation courante pour AMI
4 :     β : Meilleur évaluation courante pour ENNEMI

5 :     Si EstFeuille(etat) Alors
6 :         Retourner evalue(etat)                    ▷ Évaluation heuristique
7 :     Fin Si
8 :     Pour Tout successeur s de etat Faire
9 :         β ← min(β,MaxValue(s, α, β))
10 :        Si α ≥ β Alors                            ▷ Coupe α
11 :            Retourner α
12 :        Fin Si
13 :    Fin Pour
```

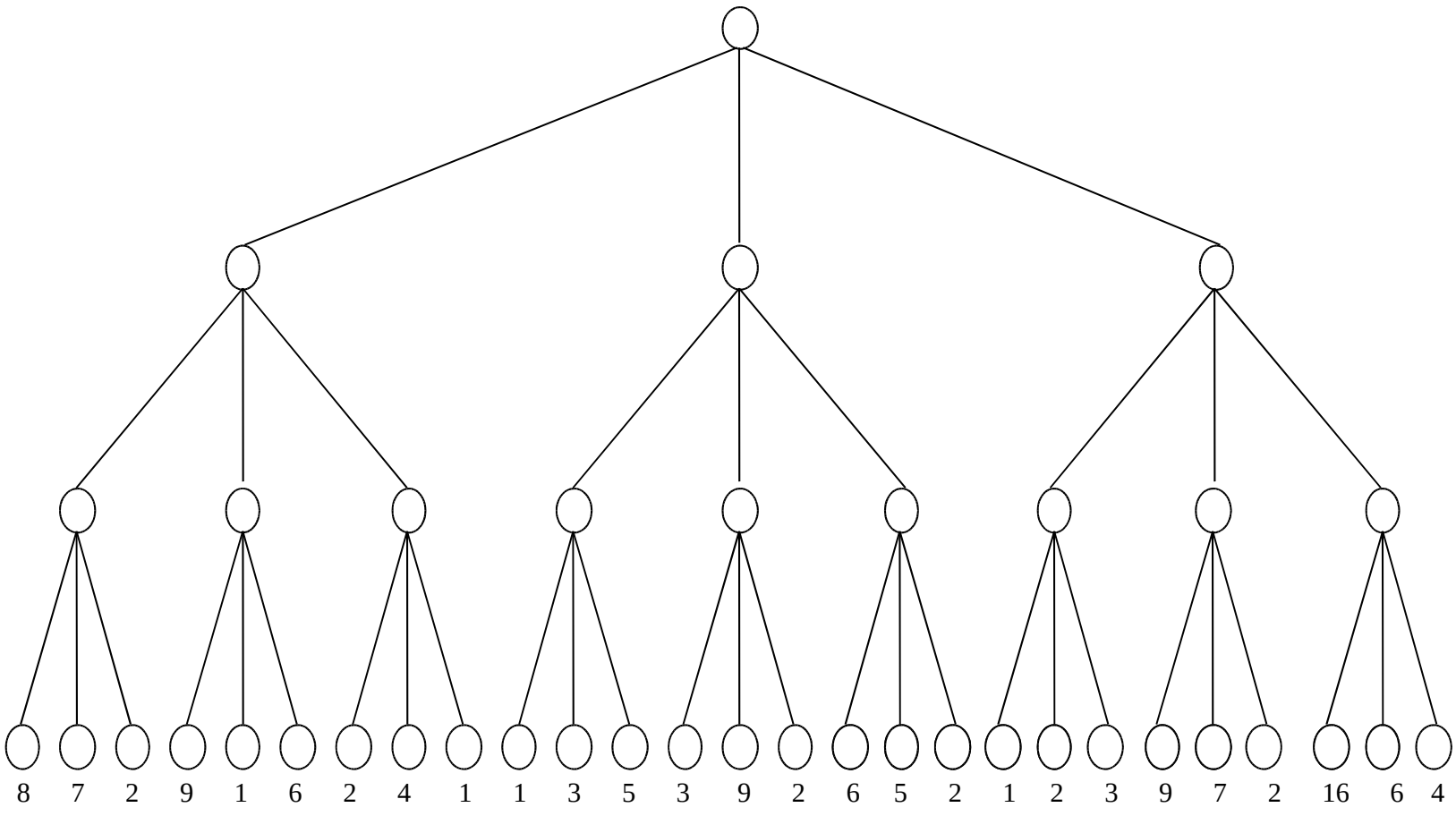
Algorithme $\alpha\beta$

Les deux fonctions ensembles

```
Fonction MaxValue(etat, α, β))                          ▷ Niveau AMI
    Si EstFeuille(etat) Alors
        Retourner evalue(etat)                          ▷ Évaluation heuristique
    Fin Si
    Pour Tout successeur s de etat Faire
        α ← max(α,MinValue(s, α, β))
        Si α ≥ β Alors                                  ▷ Coupe β
            Retourner β
        Fin Si
    Fin Pour
    Retourner α
Fin Fonction
```

```
Fonction MinValue(etat, α, β))
    Si EstFeuille(etat) Alors
        Retourner evalue(etat)                          ▷ Niveau ENNEMI
    Fin Si
    Pour Tout successeur s de etat Faire
        β ← min(β,MaxValue(s, α, β))
        Si α ≥ β Alors                                  ▷ Coupe α
            Retourner α
        Fin Si
    Fin Pour
    Retourner β
Fin Fonction
```

Exemple d'arbre de jeu



Propriétés de $\alpha\beta$

Efficacité théorique

Si on suppose que les fils sont ordonnées idéalement (ou presque), le coût de l'exploration est de $O(b^{d/2})$ au lieu de $O(b^d)$

- Le facteur de branchement théorique passe de b à $\sqrt{b}$
- La recherche peut aller deux fois plus loin dans l'arbre

Améliorations possibles

- Comment couper encore plus lors de la recherche ?
  - Recherche aspirante : α et β sont initialisés.
  - Si la valeur finale est bien dans $[\alpha, \beta]$ la recherche reste admissible
- ... D'autres encore à venir !

étude de α-β

Négamax

Une réécriture simple de MiniMax

Idée

Au lieu d'alterner deux fonctions Max/Min, on se restreint à une seule fonction en utilisant l'opposé du résultat à chaque niveau.

Attention

Cette famille d'algorithmes ne fonctionne que pour les niveaux pairs. Il faut adapter le résultat de la fonction heuristique sinon.

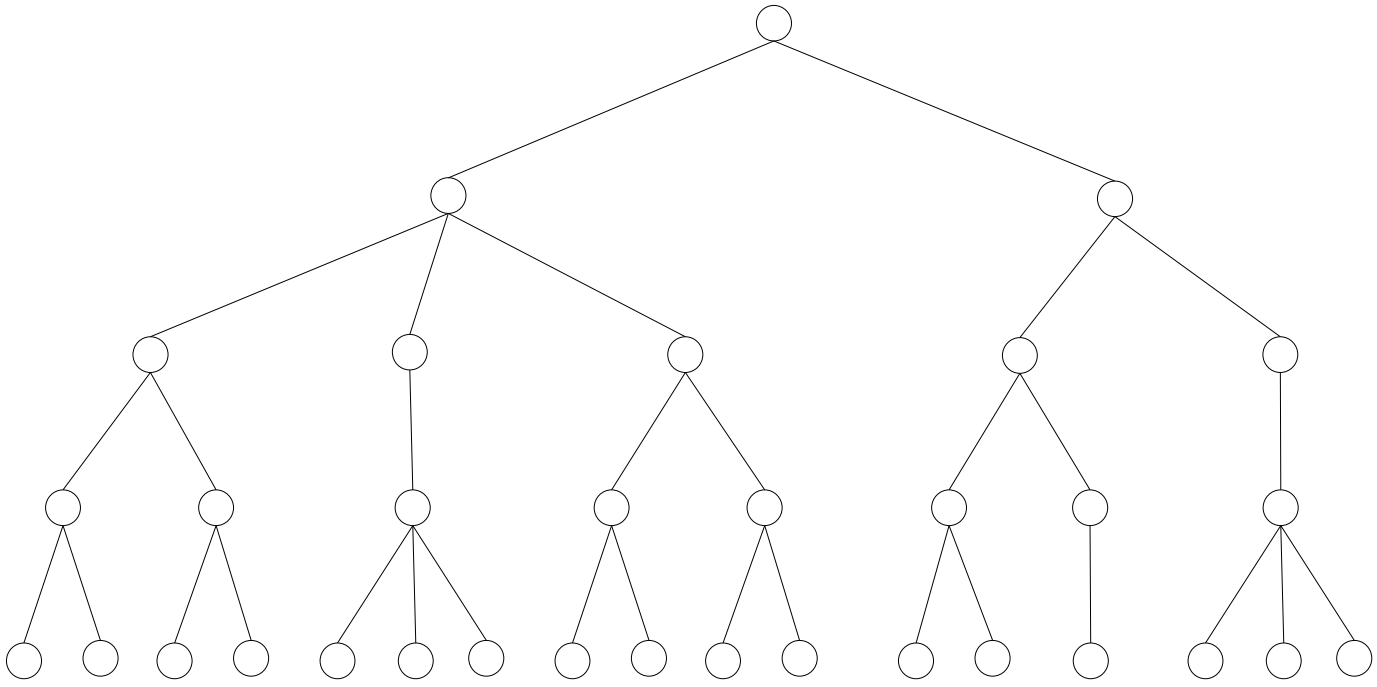
Négamax

```
1 : Fonction NegaMax(etat)                          ▷ Évaluation niveau AMI
2 :     etat : Plateau de jeu courant
3 :     Meilleur : Évaluation du meilleur coup (localement)

4 :     Si EstFeuille(etat) Alors                   ▷ Fin de partie ou horizon atteint
5 :         Retourner evalue(etat)                  ▷ Évaluation heuristique
6 :     Fin Si
7 :     Meilleur ← −∞
8 :     Pour Tout successeur s de etat Faire
9 :         val ← −NegaMax(s)
10 :        Si val ≥ Meilleur Alors
11 :            Meilleur ← val
12 :        Fin Si
13 :    Fin Pour
14 :    Retourner Meilleur
15 : Fin Fonction
```

Négamax appliqué à $\alpha\beta$

```
1 : Fonction Negαβ(etat, α, β)                          ▷ Évaluation niveau AMI

2 :     Si EstFeuille(etat) Alors                       ▷ Fin de partie ou horizon atteint
3 :         Retourner evalue(etat)                      ▷ Évaluation heuristique
4 :     Fin Si
5 :     Pour Tout successeur s de etat Faire
6 :         val ← −Negαβ(s, −β, −α)
7 :         Si val > α Alors
8 :             α ← val
9 :             Si α > β Alors
10 :                Retourner α                         ▷ Coupe
11 :            Fin Si
12 :        Fin Si
13 :    Fin Pour
14 :    Retourner α
15 : Fin Fonction
```

Exemple d'arbre de jeu avec Néga$\alpha\beta$



Quelles sont les performances de $\alpha\beta$?

Questions

- Combien de noeuds $\alpha\beta$ explore-t-il ?
- Quelle est la différence de performance avec MiniMax ?
- Avec le même temps donné quelle est la différence de profondeur entre les deux méthodes ?

Definition

On appelle arbre de jeu **uniforme de largeur** l, un arbre de jeu où tous les noeuds non terminaux ont exactement l fils.

MiniMax explore donc exactement l^p noeuds, où p est la profondeur de recherche.
Combien de noeuds $\alpha\beta$ explore-t-il ?

Quelles sont les performances de $\alpha\beta$?

Questions

- Combien de noeuds $\alpha\beta$ explore-t-il ?
- Quelle est la différence de performance avec MiniMax ?
- Avec le même temps donné quelle est la différence de profondeur entre les deux méthodes ?

Definition

On appelle arbre de jeu **uniforme de largeur** l, un arbre de jeu où tous les noeuds non terminaux ont exactement l fils.

MiniMax explore donc exactement l^p noeuds, où p est la profondeur de recherche.
Combien de noeuds $\alpha\beta$ explore-t-il ?

Quelles sont les performances de $\alpha\beta$?

Questions

- Combien de noeuds $\alpha\beta$ explore-t-il ?
- Quelle est la différence de performance avec MiniMax ?
- Avec le même temps donné quelle est la différence de profondeur entre les deux méthodes ?

Definition

On appelle arbre de jeu **uniforme de largeur** l, un arbre de jeu où tous les noeuds non terminaux ont exactement l fils.

MiniMax explore donc exactement l^p noeuds, où p est la profondeur de recherche.

Combien de noeuds $\alpha\beta$ explore-t-il ?

Quelles sont les performances de $\alpha\beta$?

Questions

- Combien de noeuds $\alpha\beta$ explore-t-il ?
- Quelle est la différence de performance avec MiniMax ?
- Avec le même temps donné quelle est la différence de profondeur entre les deux méthodes ?

Definition

On appelle arbre de jeu **uniforme de largeur** l, un arbre de jeu où tous les noeuds non terminaux ont exactement l fils.

MiniMax explore donc exactement l^p noeuds, où p est la profondeur de recherche.
Combien de noeuds $\alpha\beta$ explore-t-il ?

Types de noeuds visités lors de l'exploration

Tous les noeuds élagués lors de la recherche $Nega\alpha\beta$ le sont dès que $\alpha \geq \beta$. L'appel récursif $Nega\alpha\beta$ se ferait sur la fenêtre $[\alpha, \beta]$ avec $\alpha \geq \beta$.

Trois types de noeuds jamais élagués

À chaque niveau de l'arbre, $\alpha\beta$ est appelé avec une certaine fenêtre d'appel $[\alpha, \beta]$. Trois types de noeuds ne *peuvent donc* jamais être élagués.

1. la fenêtre d'appel est $[-\infty, +\infty]$
2. la fenêtre d'appel est $[-\infty, b]$ avec $b \neq +\infty$
3. la fenêtre d'appel est $[a, +\infty]$ avec $a \neq -\infty$

Note 1 : c'est en supposant que ∞ n'est pas une valeur heuristique.
Note 2 : les fils d'un noeud non élagué peuvent bien entendu l'être.

Types de noeuds visités lors de l'exploration

Tous les noeuds élagués lors de la recherche $Nega\alpha\beta$ le sont dès que $\alpha \geq \beta$. L'appel récursif $Nega\alpha\beta$ se ferait sur la fenêtre $[\alpha, \beta]$ avec $\alpha \geq \beta$.

Trois types de noeuds jamais élagués

À chaque niveau de l'arbre, $\alpha\beta$ est appelé avec une certaine fenêtre d'appel $[\alpha, \beta]$. Trois types de noeuds ne *peuvent donc* jamais être élagués.

1. la fenêtre d'appel est $[-\infty, +\infty]$
2. la fenêtre d'appel est $[-\infty, b]$ avec $b \neq +\infty$
3. la fenêtre d'appel est $[a, +\infty]$ avec $a \neq -\infty$

Note 1 : c'est en supposant que ∞ n'est pas une valeur heuristique.
Note 2 : les fils d'un noeud non élagué peuvent bien entendu l'être.

Types de noeuds visités lors de l'exploration

Tous les noeuds élagués lors de la recherche $Nega\alpha\beta$ le sont dès que $\alpha \geq \beta$. L'appel récursif $Nega\alpha\beta$ se ferait sur la fenêtre $[\alpha, \beta]$ avec $\alpha \geq \beta$.

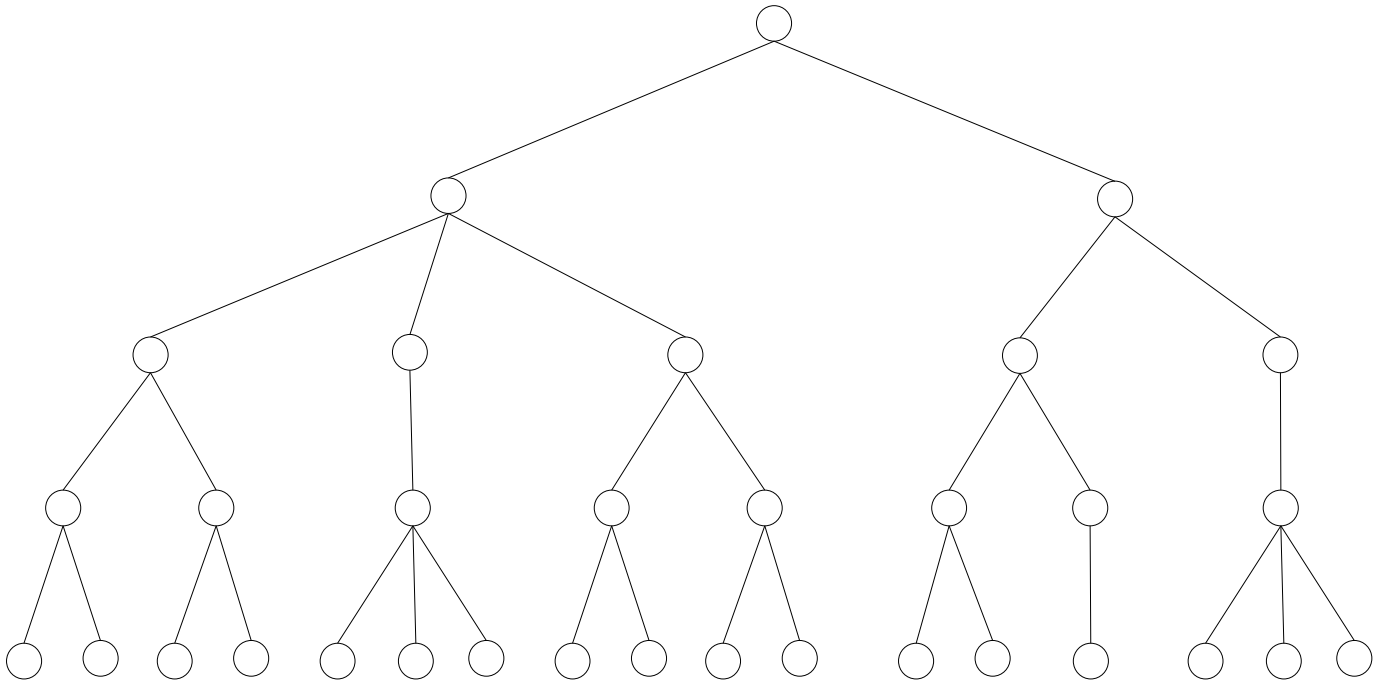
Trois types de noeuds jamais élagués

À chaque niveau de l'arbre, $\alpha\beta$ est appelé avec une certaine fenêtre d'appel $[\alpha, \beta]$. Trois types de noeuds ne *peuvent donc* jamais être élagués.

1. la fenêtre d'appel est $[-\infty, +\infty]$
2. la fenêtre d'appel est $[-\infty, b]$ avec $b \neq +\infty$
3. la fenêtre d'appel est $[a, +\infty]$ avec $a \neq -\infty$

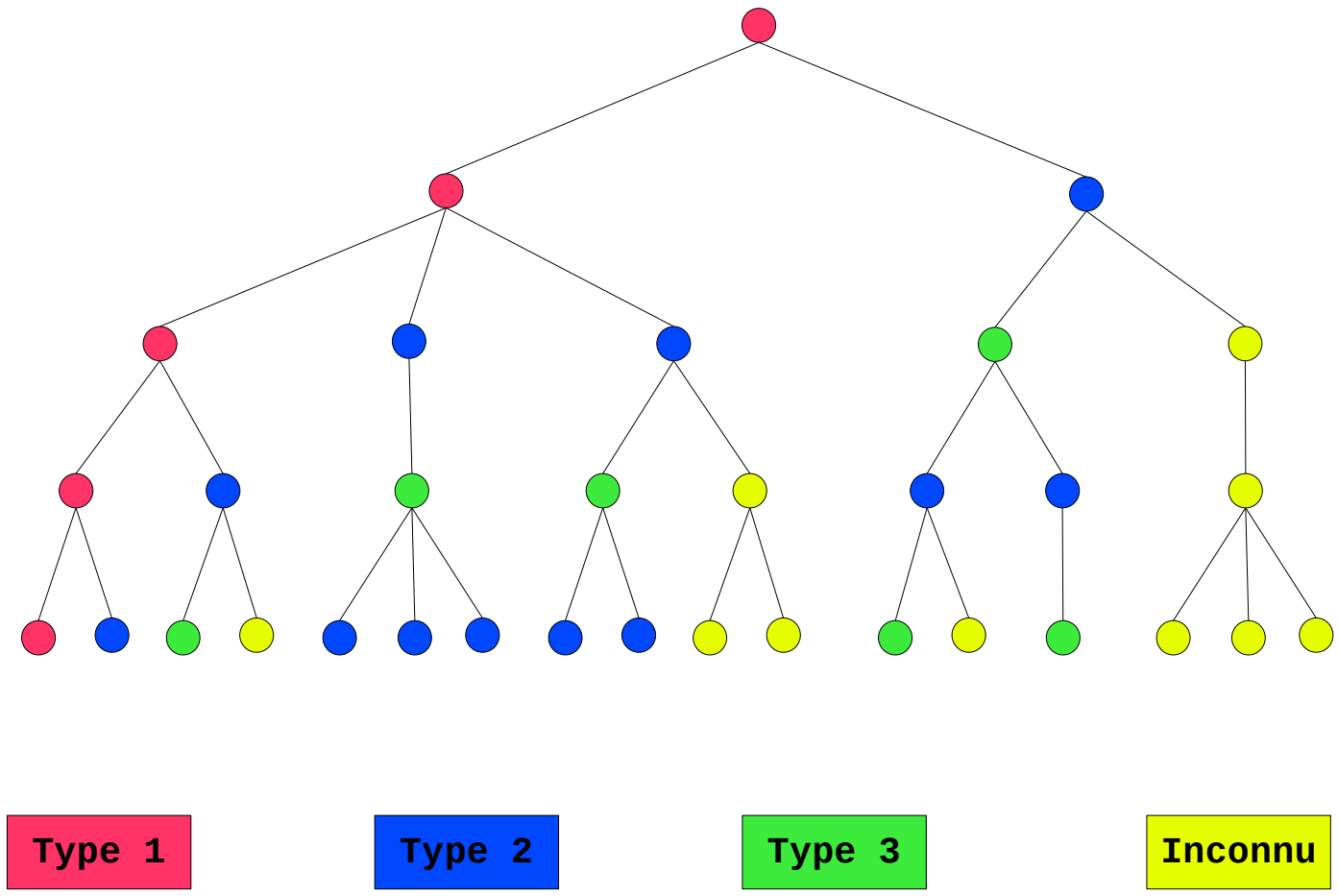
Note 1 : c'est en supposant que ∞ n'est pas une valeur heuristique.
Note 2 : les fils d'un noeud non élagué peuvent bien entendu l'être.

Types de noeuds : exemple



Type 1 **Type 2** **Type 3** **Inconnu**

Types de noeuds : exemple



Types de noeuds visités par $\alpha\beta$

Visite de l'arbre critique

L'algorithme $\alpha\beta$ utilisé avec la fenêtre $[-\infty, +\infty]$ regarde au moins *l'arbre critique*, c'est-à-dire l'ensemble des noeuds de types 1, 2 et 3, et uniquement celui-ci dans le cas où l'arbre est parfaitement ordonné.

Si l'arbre n'est pas parfaitement ordonné, on visitera plus de noeuds, jusqu'à l'arbre complet (l^p noeuds).
Quelle est la borne inférieur ?

Types de noeuds visités par $\alpha\beta$

Visite de l'arbre critique

L'algorithme $\alpha\beta$ utilisé avec la fenêtre $[-\infty, +\infty]$ regarde au moins *l'arbre critique*, c'est-à-dire l'ensemble des noeuds de types 1, 2 et 3, et uniquement celui-ci dans le cas où l'arbre est parfaitement ordonné.

Si l'arbre n'est pas parfaitement ordonné, on visitera plus de noeuds, jusqu'à l'arbre complet (l^p noeuds).
Quelle est la borne inférieur ?

Types de noeuds visités par $\alpha\beta$

Visite de l'arbre critique

L'algorithme $\alpha\beta$ utilisé avec la fenêtre $[-\infty, +\infty]$ regarde au moins *l'arbre critique*, c'est-à-dire l'ensemble des noeuds de types 1, 2 et 3, et uniquement celui-ci dans le cas où l'arbre est parfaitement ordonné.

Si l'arbre n'est pas parfaitement ordonné, on visitera plus de noeuds, jusqu'à l'arbre complet (l^p noeuds).

Quelle est la borne inférieur ?

Types de noeuds visités par $\alpha\beta$

Visite de l'arbre critique

L'algorithme $\alpha\beta$ utilisé avec la fenêtre $[-\infty, +\infty]$ regarde au moins *l'arbre critique*, c'est-à-dire l'ensemble des noeuds de types 1, 2 et 3, et uniquement celui-ci dans le cas où l'arbre est parfaitement ordonné.

Si l'arbre n'est pas parfaitement ordonné, on visitera plus de noeuds, jusqu'à l'arbre complet (l^p noeuds).
Quelle est la borne inférieur ?

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :

$u_0 = 0,\ v_0 = 0$

$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$

Démonstration :

$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$

$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$

$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$$

$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence

Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.

Suppositions :

$u_0 = 0,\ v_0 = 0$

$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$

Démonstration :

$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$

$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$

$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$$

$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :
$u_0 = 0,\ v_0 = 0$
$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$
Démonstration :
$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$
$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$
$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$$
$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :
$u_0 = 0,\ v_0 = 0$
$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$
Démonstration :
$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$
$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$
$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$
$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :

$u_0 = 0,\ v_0 = 0$

$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$

Démonstration :

$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$

$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$

$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$$

$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :

$u_0 = 0,\ v_0 = 0$

$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$

Démonstration :

$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$

$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$

$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$$

$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :

$u_0 = 0,\ v_0 = 0$

$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$

Démonstration :

$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$

$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$

$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$$

$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :

$u_0 = 0,\ v_0 = 0$

$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$

Démonstration :

$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$

$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$

$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$$

$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

Preuve par récurrence
Soit u_n le nombre de feuilles de type 2,
v_n le nombre de feuilles de type 2 pour un arbre critique uniforme de largeur l et de profondeur n.
Suppositions :

$u_0 = 0,\ v_0 = 0$

$\forall n > 0,\ u_n = lv_{n-1} + l - 1,\ v_n = u_{n-1}$

Démonstration :

$\Rightarrow \forall n > 2,\ u_n = lu_{n-2} + l - 1$

$\Rightarrow \forall n > 2,\ u_n + 1 = l(u_{n-2} + 1)$

$$\Rightarrow \forall n > 2,\ u_n = \begin{cases} l^{n/2} - 1 & \text{Si } n \text{ est pair} \\ l^{(n-1)/2}l - 1 & \text{Si } n \text{ est impair} \end{cases}$$

$\Rightarrow \forall n,\ u_n = l^{n/2} - 1$

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

En résumé

On a donc compté

- $l^{n/2} - 1$ noeuds de type 2
- d'où $l^{(n-1)/2} - 1$ noeuds de type 3
- et 1 noeud de type 1

Soit, au mieux pour $\alpha\beta$ de l'ordre de :

$$l^{n/2} - 1 + l^{(n-1)/2} - 1 + 1$$

ou encore

$$l^{n/2}$$

noeuds

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

En résumé

On a donc compté

- $l^{n/2} - 1$ noeuds de type 2
- d'où $l^{(n-1)/2} - 1$ noeuds de type 3
- et 1 noeud de type 1

Soit, au mieux pour $\alpha\beta$ de l'ordre de :

$$l^{n/2} - 1 + l^{(n-1)/2} - 1 + 1$$

ou encore

$$l^{n/2}$$

noeuds

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

En résumé

On a donc compté

- $l^{n/2} - 1$ noeuds de type 2
- d'où $l^{(n-1)/2} - 1$ noeuds de type 3
- et 1 noeud de type 1

Soit, au mieux pour $\alpha\beta$ de l'ordre de :

$$l^{n/2} - 1 + l^{(n-1)/2} - 1 + 1$$

ou encore

$$l^{n/2}$$

noeuds

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

En résumé

On a donc compté

- $l^{n/2} - 1$ noeuds de type 2
- d'où $l^{(n-1)/2} - 1$ noeuds de type 3
- et 1 noeud de type 1

Soit, au mieux pour $\alpha\beta$ de l'ordre de :

$$l^{n/2} - 1 + l^{(n-1)/2} - 1 + 1$$

ou encore

$$l^{n/2}$$

noeuds

Comptage des différents noeuds

Types de noeuds :

- type 1 : racine et fils ainés des type 1
- type 2 : cadets de type 1 et tous les fils de type 3
- type 3 : fils aînés des type 2
- inconnus : cadets des type 2 et les fils des inconnus

En résumé

On a donc compté

- $l^{n/2} - 1$ noeuds de type 2
- d'où $l^{(n-1)/2} - 1$ noeuds de type 3
- et 1 noeud de type 1

Soit, au mieux pour $\alpha\beta$ de l'ordre de :

$$l^{n/2} - 1 + l^{(n-1)/2} - 1 + 1$$

ou encore

$$l^{n/2}$$

noeuds

Résultats des performances

Encadrement des performances de $\alpha\beta$

Le nombre de feuilles évaluées par $\alpha\beta$ est compris entre $l^{p/2}$ et l^p pour un arbre de largeur l et de profondeur p

Nombre de noeuds visités

$\alpha\beta$ visite au minimum un nombre de noeuds de l'ordre de $l^{p/2}$

Comparaison MiniMax / $\alpha\beta$

Au pire, $\alpha\beta$ explore tous les noeuds de l'arbre. Au mieux, $\alpha\beta$ peut voir à un horizon deux fois plus lointain que MiniMax dans le même temps.

L'ordre de développement des fils d'un noeud est primordial pour obtenir de bonnes performances !

Résultats des performances

Encadrement des performances de $\alpha\beta$

Le nombre de feuilles évaluées par $\alpha\beta$ est compris entre $l^{p/2}$ et l^p pour un arbre de largeur l et de profondeur p

Nombre de noeuds visités

$\alpha\beta$ visite au minimum un nombre de noeuds de l'ordre de $l^{p/2}$

Comparaison MiniMax / $\alpha\beta$

Au pire, $\alpha\beta$ explore tous les noeuds de l'arbre. Au mieux, $\alpha\beta$ peut voir à un horizon deux fois plus lointain que MiniMax dans le même temps.

L'ordre de développement des fils d'un noeud est primordial pour obtenir de bonnes performances !

Résultats des performances

Encadrement des performances de $\alpha\beta$

Le nombre de feuilles évaluées par $\alpha\beta$ est compris entre $l^{p/2}$ et l^p pour un arbre de largeur l et de profondeur p

Nombre de noeuds visités

$\alpha\beta$ visite au minimum un nombre de noeuds de l'ordre de $l^{p/2}$

Comparaison MiniMax / $\alpha\beta$

Au pire, $\alpha\beta$ explore tous les noeuds de l'arbre. Au mieux, $\alpha\beta$ peut voir à un horizon deux fois plus lointain que MiniMax dans le même temps.

L'ordre de développement des fils d'un noeud est primordial pour obtenir de bonnes performances !

Résultats des performances

Encadrement des performances de $\alpha\beta$

Le nombre de feuilles évaluées par $\alpha\beta$ est compris entre $l^{p/2}$ et l^p pour un arbre de largeur l et de profondeur p

Nombre de noeuds visités

$\alpha\beta$ visite au minimum un nombre de noeuds de l'ordre de $l^{p/2}$

Comparaison MiniMax / $\alpha\beta$

Au pire, $\alpha\beta$ explore tous les noeuds de l'arbre. Au mieux, $\alpha\beta$ peut voir à un horizon deux fois plus lointain que MiniMax dans le même temps.

L'ordre de développement des fils d'un noeud est primordial pour obtenir de bonnes performances !

Résultats des performances

Encadrement des performances de $\alpha\beta$

Le nombre de feuilles évaluées par $\alpha\beta$ est compris entre $l^{p/2}$ et l^p pour un arbre de largeur l et de profondeur p

Nombre de noeuds visités

$\alpha\beta$ visite au minimum un nombre de noeuds de l'ordre de $l^{p/2}$

Comparaison MiniMax / $\alpha\beta$

Au pire, $\alpha\beta$ explore tous les noeuds de l'arbre. Au mieux, $\alpha\beta$ peut voir à un horizon deux fois plus lointain que MiniMax dans le même temps.

L'ordre de développement des fils d'un noeud est primordial pour obtenir de bonnes performances !

À la recherche de performances

Problématique du temps réel

Questions liées au choix de la profondeur de recherche

La profondeur de recherche doit être fixée avant de lancer la recherche (L'horizon doit être équilibré sur tout l'arbre de recherche). Comment choisir la profondeur ?

- Comment garantir au joueur que l'ordinateur ne va pas passer trop de temps à réflechir ?
- Comment le garantir avec des machines différentes ?
- Comment lier le temps passé à réflechir avec la profondeur maximale de l'horizon ?
- Que se passe-t-il si l'arbre n'est pas homogène (hypothèse (très) réaliste) ?

Comment jouer au mieux dans un temps donné ?

Problématique du temps réel

Questions liées au choix de la profondeur de recherche

La profondeur de recherche doit être fixée avant de lancer la recherche (L'horizon doit être équilibré sur tout l'arbre de recherche). Comment choisir la profondeur ?

- Comment garantir au joueur que l'ordinateur ne va pas passer trop de temps à réflechir ?
- Comment le garantir avec des machines différentes ?
- Comment lier le temps passé à réflechir avec la profondeur maximale de l'horizon ?
- Que se passe-t-il si l'arbre n'est pas homogène (hypothèse (très) réaliste) ?

Comment jouer au mieux dans un temps donné ?

Problématique du temps réel

Questions liées au choix de la profondeur de recherche

La profondeur de recherche doit être fixée avant de lancer la recherche (L'horizon doit être équilibré sur tout l'arbre de recherche). Comment choisir la profondeur ?

- Comment garantir au joueur que l'ordinateur ne va pas passer trop de temps à réflechir ?
- Comment le garantir avec des machines différentes ?
- Comment lier le temps passé à réflechir avec la profondeur maximale de l'horizon ?
- Que se passe-t-il si l'arbre n'est pas homogène (hypothèse (très) réaliste) ?

Comment jouer au mieux dans un temps donné ?

Problématique du temps réel

Questions liées au choix de la profondeur de recherche

La profondeur de recherche doit être fixée avant de lancer la recherche (L'horizon doit être équilibré sur tout l'arbre de recherche). Comment choisir la profondeur ?

- Comment garantir au joueur que l'ordinateur ne va pas passer trop de temps à réflechir ?
- Comment le garantir avec des machines différentes ?
- Comment lier le temps passé à réflechir avec la profondeur maximale de l'horizon ?
- Que se passe-t-il si l'arbre n'est pas homogène (hypothèse (très) réaliste) ?

Comment jouer au mieux dans un temps donné ?

Iterative Deepening

Maîtriser le temps

Idée : Étendre peu à peu l'horizon de $\alpha\beta$

Mais doit-on mémoriser tout l'arbre pour repartir des feuilles ?

Iterative Deepening (ID)

Soit n initialisé à un horizon *immédiat*.

- Faire une recherche à horizon n
- S'il reste du temps, tout oublier (sauf le coup à jouer) et incrémenter n.

Et les performances ?

ID semble refaire beaucoup de fois la même chose. Est-ce pour autant très inefficace ?

Iterative Deepening

Maîtriser le temps

Idée : Étendre peu à peu l'horizon de $\alpha\beta$

Mais doit-on mémoriser tout l'arbre pour repartir des feuilles ?

Iterative Deepening (ID)

Soit n initialisé à un horizon *immédiat*.

- Faire une recherche à horizon n
- S'il reste du temps, tout oublier (sauf le coup à jouer) et incrémenter n.

Et les performances ?

ID semble refaire beaucoup de fois la même chose. Est-ce pour autant très inefficace ?

Iterative Deepening

Maîtriser le temps

Idée : Étendre peu à peu l'horizon de $\alpha\beta$

Mais doit-on mémoriser tout l'arbre pour repartir des feuilles ?

Iterative Deepening (ID)

Soit n initialisé à un horizon *immédiat*.

- Faire une recherche à horizon n
- S'il reste du temps, tout oublier (sauf le coup à jouer) et incrémenter n.

Et les performances ?

ID semble refaire beaucoup de fois la même chose. Est-ce pour autant très inefficace ?

Iterative Deepening

Maîtriser le temps

Idée : Étendre peu à peu l'horizon de $\alpha\beta$

Mais doit-on mémoriser tout l'arbre pour repartir des feuilles ?

Iterative Deepening (ID)

Soit n initialisé à un horizon *immédiat*.

- Faire une recherche à horizon n
- S'il reste du temps, tout oublier (sauf le coup à jouer) et incrémenter n.

Et les performances ?

ID semble refaire beaucoup de fois la même chose. Est-ce pour autant très inefficace ?

Iterative Deepening

Un algorithme pas si *inefficace*

La croissance exponentielle des arbres montre intuitivement que la majorité des noeuds (donc de la difficulté) se situent sur les feuilles...
Développer le dernier niveau coûte le plus cher, d'autant plus si on a un grand facteur de branchement.

Qui veut des chiffres ?

Une recherche à profondeur d et facteur de branchement b coûte

$$1 + b + b^2 + \ldots + b^{d-1} + b^d$$

Pour $(d = 5, b = 10)$ on a 11 111 noeuds. Si on fait un ID, on aurait développé 12 345 noeuds au total (1 pour d=1, 11 pour d=2, 111 pour d=3, 1 111 pour d=4 ...), soit 11% de noeuds supplémentaires (seulement).

Iterative Deepening

Un algorithme pas si *inefficace*

La croissance exponentielle des arbres montre intuitivement que la majorité des noeuds (donc de la difficulté) se situent sur les feuilles... Développer le dernier niveau coûte le plus cher, d'autant plus si on a un grand facteur de branchement.

Qui veut des chiffres ?

Une recherche à profondeur d et facteur de branchement b coûte

$$1 + b + b^2 + \ldots + b^{d-1} + b^d$$

Pour $(d = 5, b = 10)$ on a 11 111 noeuds. Si on fait un ID, on aurait développé 12 345 noeuds au total (1 pour d=1, 11 pour d=2, 111 pour d=3, 1 111 pour d=4 ...), soit 11% de noeuds supplémentaires (seulement).

Iterative Deepening

Un algorithme pas si *inefficace*

La croissance exponentielle des arbres montre intuitivement que la majorité des noeuds (donc de la difficulté) se situent sur les feuilles... Développer le dernier niveau coûte le plus cher, d'autant plus si on a un grand facteur de branchement.

Qui veut des chiffres ?

Une recherche à profondeur d et facteur de branchement b coûte

$$1 + b + b^2 + \ldots + b^{d-1} + b^d$$

Pour $(d = 5, b = 10)$ on a 11 111 noeuds. Si on fait un ID, on aurait développé 12 345 noeuds au total (1 pour d=1, 11 pour d=2, 111 pour d=3, 1 111 pour d=4 ...), soit 11% de noeuds supplémentaires (seulement).

Iterative Deepening

Un algorithme pas si *inefficace*

La croissance exponentielle des arbres montre intuitivement que la majorité des noeuds (donc de la difficulté) se situent sur les feuilles... Développer le dernier niveau coûte le plus cher, d'autant plus si on a un grand facteur de branchement.

Qui veut des chiffres ?

Une recherche à profondeur d et facteur de branchement b coûte

$$1 + b + b^2 + \ldots + b^{d-1} + b^d$$

Pour $(d = 5, b = 10)$ on a 11 111 noeuds. Si on fait un ID, on aurait développé 12 345 noeuds au total (1 pour d=1, 11 pour d=2, 111 pour d=3, 1 111 pour d=4 ...), soit 11% de noeuds supplémentaires (seulement).

Iterative Deepening

Des avantages de poids avec $\alpha\beta$

Avantages

- Grande souplesse dans la gestion du temps. Le programme peut donner la meilleur valeur trouvée à n'importe quel moment.
- Couplé à $\alpha\beta$, ID peut même devenir plus efficace qu'un seul $\alpha\beta$

Comment ?

$\alpha\beta$ élague d'autant plus que les meilleurs coups sont développés en premier. On profite donc de l'ancien $\alpha\beta$ à profondeur n pour ordonner les fils à profondeur $n+1$.

Iterative Deepening

Des avantages de poids avec $\alpha\beta$

Avantages

- Grande souplesse dans la gestion du temps. Le programme peut donner la meilleur valeur trouvée à n'importe quel moment.
- Couplé à $\alpha\beta$, ID peut même devenir plus efficace qu'un seul $\alpha\beta$

Comment ?

$\alpha\beta$ élague d'autant plus que les meilleurs coups sont développés en premier. On profite donc de l'ancien $\alpha\beta$ à profondeur n pour ordonner les fils à profondeur $n + 1$.

Iterative Deepening

En pratique

Idées

Plutôt que de tout oublier après chaque relancement de ID, on va essayer de mémoriser pour réordonner l'arbre ensuite.

Première méthode

Garder tout l'arbre de recherche en mémoire, l'ordonner à chaque niveau puis relancer $\alpha\beta$ directement sur le nouvel arbre *optimal*.

Seconde méthode

Réserver une zone mémoire pour associer à chaque plateau de jeu déjà vu le meilleur des fils et la valeur heuristique associée (ainsi que sa profondeur).

Iterative Deepening

En pratique

Idées

Plutôt que de tout oublier après chaque relancement de ID, on va essayer de mémoriser pour réordonner l'arbre ensuite.

Première méthode

Garder tout l'arbre de recherche en mémoire, l'ordonner à chaque niveau puis relancer $\alpha\beta$ directement sur le nouvel arbre *optimal*.

Seconde méthode

Réserver une zone mémoire pour associer à chaque plateau de jeu déjà vu le meilleur des fils et la valeur heuristique associée (ainsi que sa profondeur).

Iterative Deepening

En pratique

Idées

Plutôt que de tout oublier après chaque relancement de ID, on va essayer de mémoriser pour réordonner l'arbre ensuite.

Première méthode

Garder tout l'arbre de recherche en mémoire, l'ordonner à chaque niveau puis relancer $\alpha\beta$ directement sur le nouvel arbre *optimal*.

Seconde méthode

Réserver une zone mémoire pour associer à chaque plateau de jeu déjà vu le meilleur des fils et la valeur heuristique associée (ainsi que sa profondeur).

Iterative Deepening

En pratique

Idées

Plutôt que de tout oublier après chaque relancement de ID, on va essayer de mémoriser pour réordonner l'arbre ensuite.

Première méthode

Garder tout l'arbre de recherche en mémoire, l'ordonner à chaque niveau puis relancer $\alpha\beta$ directement sur le nouvel arbre *optimal*.

Seconde méthode

Réserver une zone mémoire pour associer à chaque plateau de jeu déjà vu le meilleur des fils et la valeur heuristique associée (ainsi que sa profondeur).

Repérer des coups profonds

Question

Comment repérer efficacement d'éventuelles stratégies gagnantes au delà de l'horizon ?

Idée

Il faut élaguer plus encore dans $\alpha\beta$. On réduit la valeur heuristique à $[0, 1]$ pour couper plus et aller plus loin. Découper le temps T que l'on a pour un coup en 3 :

- Chercher un coup gagnant profond par un $\alpha\beta$ sur $[0, 1]$. Y passer $T/10$ secondes
- Si on ne trouve pas de coups gagnant, trouver un *bon coup* avec un $\alpha\beta$ classique en $8T/10$ secondes.
- Vérifier en $T/10$ secondes que ce coup n'est pas un coup perdant à un horizon lointain

Repérer des coups profonds

Question

Comment repérer efficacement d'éventuelles stratégies gagnantes au delà de l'horizon ?

Idée

Il faut élaguer plus encore dans $\alpha\beta$. On réduit la valeur heuristique à $[0, 1]$ pour couper plus et aller plus loin. Découper le temps T que l'on a pour un coup en 3 :

- Chercher un coup gagnant profond par un $\alpha\beta$ sur $[0, 1]$. Y passer $T/10$ secondes
- Si on ne trouve pas de coups gagnant, trouver un *bon coup* avec un $\alpha\beta$ classique en $8T/10$ secondes.
- Vérifier en $T/10$ secondes que ce coup n'est pas un coup perdant à un horizon lointain

Repérer des coups profonds

Question

Comment repérer efficacement d'éventuelles stratégies gagnantes au delà de l'horizon ?

Idée

Il faut élaguer plus encore dans $\alpha\beta$. On réduit la valeur heuristique à $[0, 1]$ pour couper plus et aller plus loin. Découper le temps T que l'on a pour un coup en 3 :

- Chercher un coup gagnant profond par un $\alpha\beta$ sur $[0, 1]$. Y passer $T/10$ secondes
- Si on ne trouve pas de coups gagnant, trouver un *bon coup* avec un $\alpha\beta$ classique en $8T/10$ secondes.
- Vérifier en $T/10$ secondes que ce coup n'est pas un coup perdant à un horizon lointain

Repérer des coups profonds

Question

Comment repérer efficacement d'éventuelles stratégies gagnantes au delà de l'horizon ?

Idée

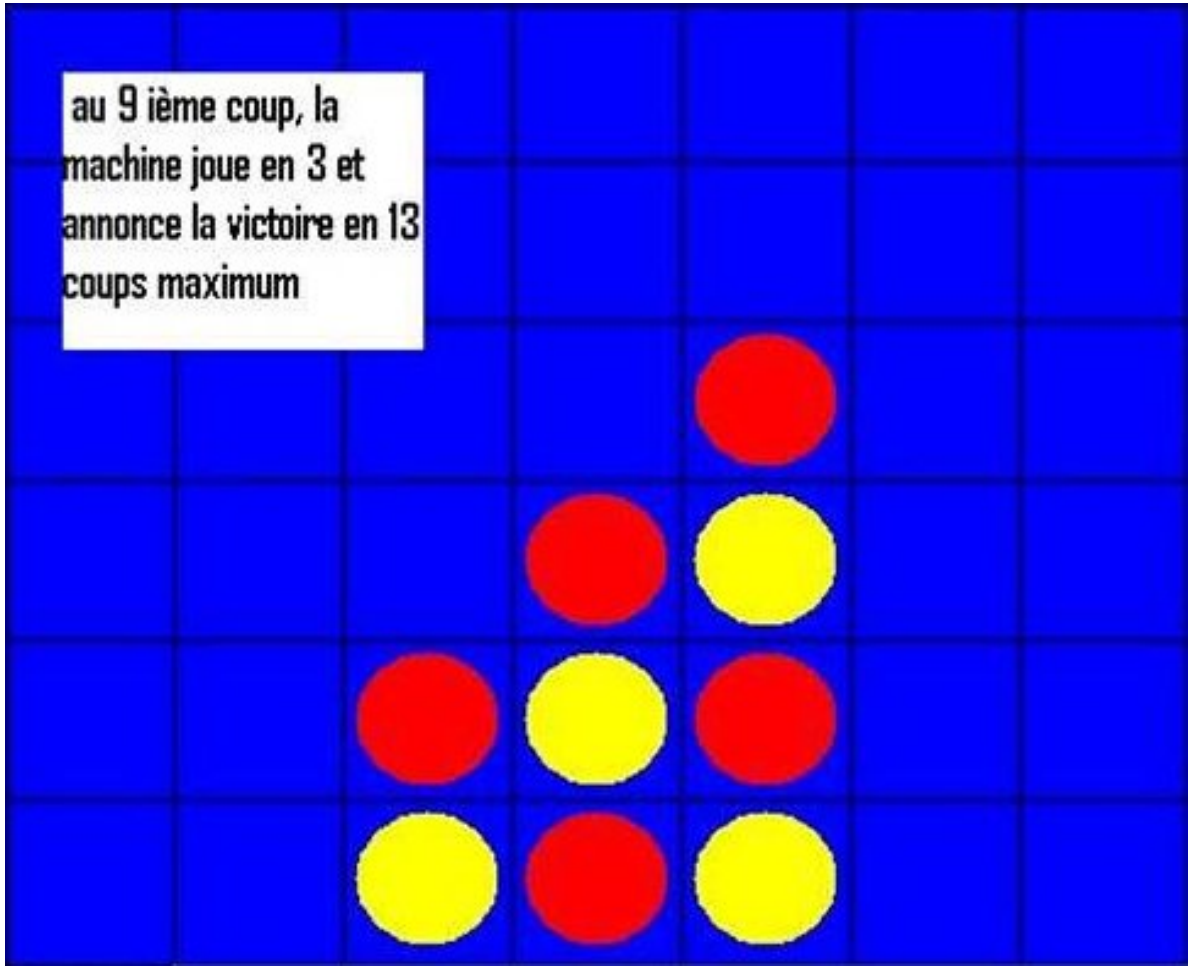
Il faut élaguer plus encore dans $\alpha\beta$. On réduit la valeur heuristique à $[0, 1]$ pour couper plus et aller plus loin. Découper le temps T que l'on a pour un coup en 3 :

- Chercher un coup gagnant profond par un $\alpha\beta$ sur $[0, 1]$. Y passer $T/10$ secondes
- Si on ne trouve pas de coups gagnant, trouver un *bon coup* avec un $\alpha\beta$ classique en $8T/10$ secondes.
- Vérifier en $T/10$ secondes que ce coup n'est pas un coup perdant à un horizon lointain

Repérer les coups fatals

Exemple sur le Puissance 4

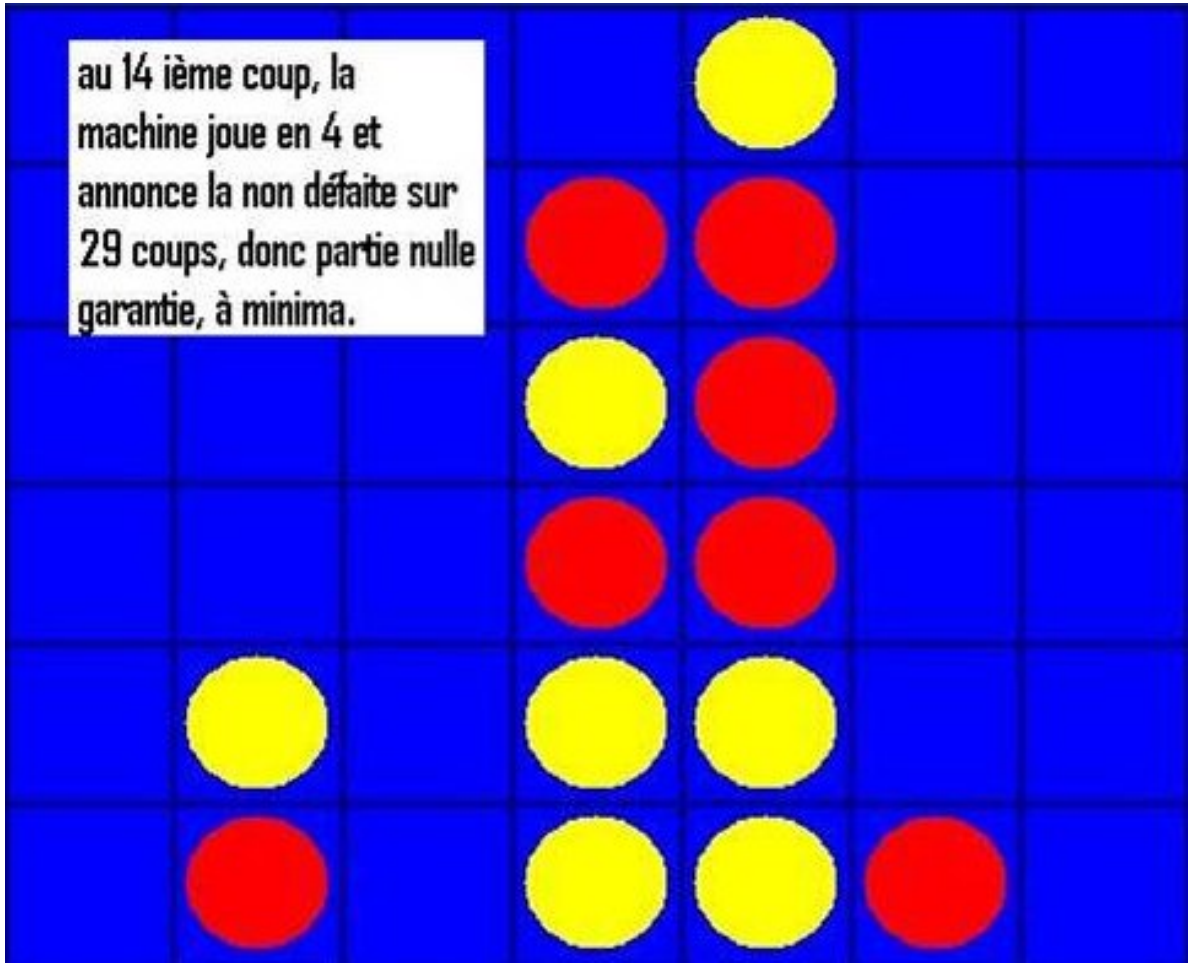
Images issues d'une applet en Java disponible sur le web.



Repérer les coups fatals

Exemple sur le Puissance 4

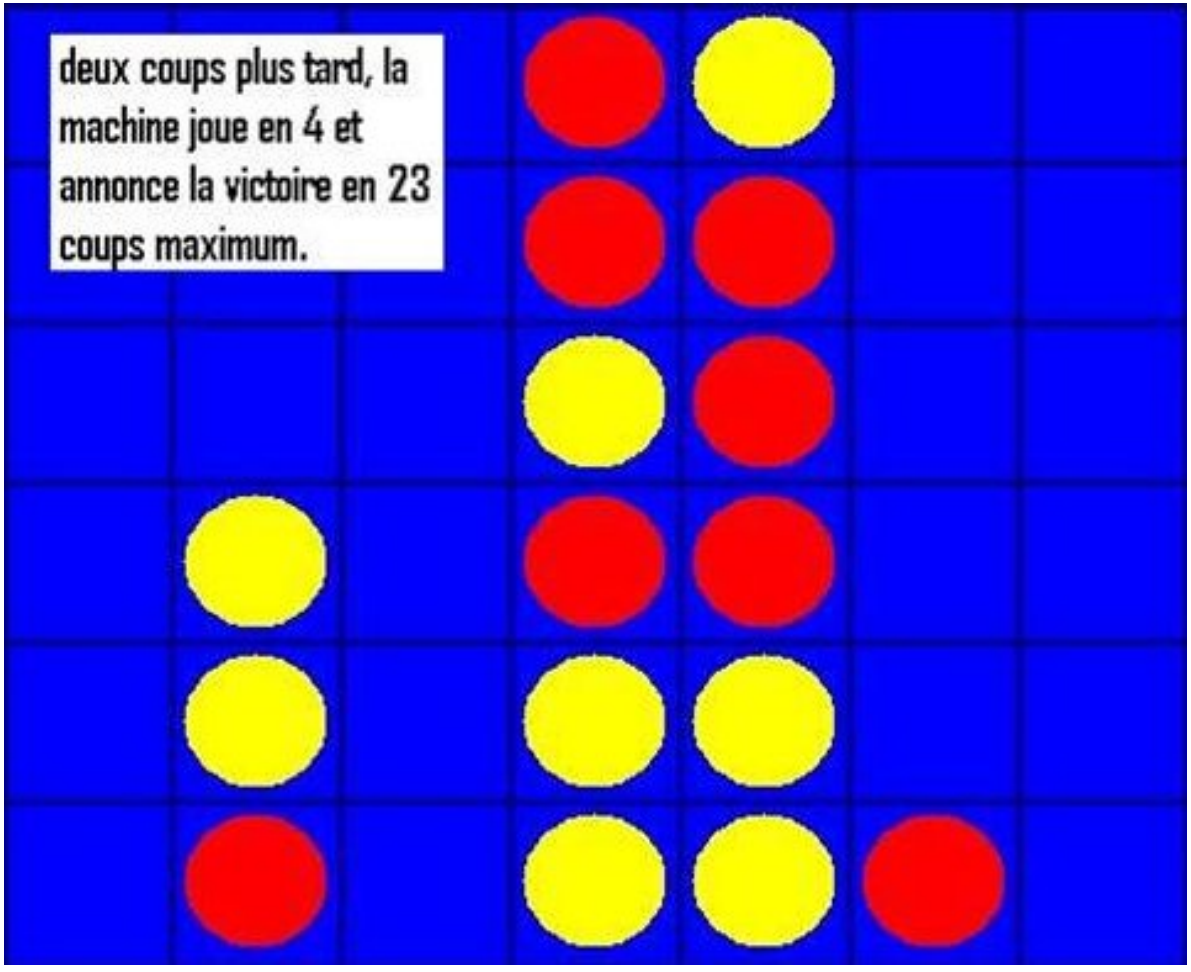
Images issues d'une applet en Java disponible sur le web.



Repérer les coups fatals

Exemple sur le Puissance 4

Images issues d'une applet en Java disponible sur le web.



Problèmes à l'horizon

Inconvénients de fixer un horizon

Tous les programmes doivent fixer un horizon. Pourtant, cela entraîne deux problèmes importants :

- Manque de clairvoyance : On ne peut prévoir un coup (même fatal) au-delà de l'horizon
- Aveuglement volontaire : l'approche MiniMax va pousser le programme à tout faire pour repousser un coup mauvais – mais inévitable – au delà de l'horizon.

Deux solutions... Parmi d'autres

- Déséquilibrer l'arbre de recherche
- Utiliser une *méta-heuristique*

Problèmes à l'horizon

Inconvénients de fixer un horizon

Tous les programmes doivent fixer un horizon. Pourtant, cela entraîne deux problèmes importants :

- Manque de clairvoyance : On ne peut prévoir un coup (même fatal) au-delà de l'horizon
- Aveuglement volontaire : l'approche MiniMax va pousser le programme à tout faire pour repousser un coup mauvais – mais inévitable – au delà de l'horizon.

Deux solutions... Parmi d'autres

- Déséquilibrer l'arbre de recherche
- Utiliser une *méta-heuristique*

Problèmes à l'horizon

Inconvénients de fixer un horizon

Tous les programmes doivent fixer un horizon. Pourtant, cela entraîne deux problèmes importants :

- Manque de clairvoyance : On ne peut prévoir un coup (même fatal) au-delà de l'horizon
- Aveuglement volontaire : l'approche MiniMax va pousser le programme à tout faire pour repousser un coup mauvais – mais inévitable – au delà de l'horizon.

Deux solutions... Parmi d'autres

- Déséquilibrer l'arbre de recherche
- Utiliser une *méta-heuristique*

Problèmes à l'horizon

Inconvénients de fixer un horizon

Tous les programmes doivent fixer un horizon. Pourtant, cela entraîne deux problèmes importants :

- Manque de clairvoyance : On ne peut prévoir un coup (même fatal) au-delà de l'horizon
- Aveuglement volontaire : l'approche MiniMax va pousser le programme à tout faire pour repousser un coup mauvais – mais inévitable – au delà de l'horizon.

Deux solutions... Parmi d'autres

- Déséquilibrer l'arbre de recherche
- Utiliser une *méta-heuristique*

Atténuation d'horizons

Zoomer où l'on va aller

Idée

Une fois que l'on a fait le choix du coup à jouer, passer un peu de temps pour voir si la fonction heuristique ne nous a pas trompé. On va déséquilibrer l'arbre de recherche.

Deux solutions... Parmi d'autres

- Redévelopper un $\alpha\beta$ sur la branche sélectionnée.
- Redévelopper un $\alpha\beta$ sur la meilleure feuille visée.
- Redévelopper un $\alpha\beta$ sur un sous-arbre *intéressant*

Atténuation d'horizons

Zoomer où l'on va aller

Idée

Une fois que l'on a fait le choix du coup à jouer, passer un peu de temps pour voir si la fonction heuristique ne nous a pas trompé. On va déséquilibrer l'arbre de recherche.

Deux solutions... Parmi d'autres

- Redévelopper un $\alpha\beta$ sur la branche sélectionnée.
- Redévelopper un $\alpha\beta$ sur la meilleure feuille visée.
- Redévelopper un $\alpha\beta$ sur un sous-arbre *intéressant*

Atténuation d'horizons

Zoomer où l'on va aller

Idée

Une fois que l'on a fait le choix du coup à jouer, passer un peu de temps pour voir si la fonction heuristique ne nous a pas trompé. On va déséquilibrer l'arbre de recherche.

Deux solutions... Parmi d'autres

- Redévelopper un $\alpha\beta$ sur la branche sélectionnée.
- Redévelopper un $\alpha\beta$ sur la meilleure feuille visée.
- Redévelopper un $\alpha\beta$ sur un sous-arbre *intéressant*

Atténuation d'horizons

Zoomer où l'on va aller

Idée

Une fois que l'on a fait le choix du coup à jouer, passer un peu de temps pour voir si la fonction heuristique ne nous a pas trompé. On va déséquilibrer l'arbre de recherche.

Deux solutions... Parmi d'autres

- Redévelopper un $\alpha\beta$ sur la branche sélectionnée.
- Redévelopper un $\alpha\beta$ sur la meilleure feuille visée.
- Redévelopper un $\alpha\beta$ sur un sous-arbre *intéressant*

Atténuation d'horizons

Reconsidérer la notion d'horizons

Idée

Définir l'horizon en fonction de l'intérêt des coups joués et non en fonction du nombre de coups.

Principes

On part par exemple avec une profondeur $SX = 10.p$, où p est la profondeur au sens habituel.

- Un coup « moyen » diminue SX de 10.
- Un coup intéressant (prise de pièce, échec) ne diminue SX que de 2 ou 3.
- Un coup peu intéressant diminue beaucoup SX (de l'ordre de 35).

Résultat : les coups intéressants seront explorés plus profondément.

Atténuation d'horizons

Reconsidérer la notion d'horizons

Idée

Définir l'horizon en fonction de l'intérêt des coups joués et non en fonction du nombre de coups.

Principes

On part par exemple avec une profondeur $SX = 10.p$, où p est la profondeur au sens habituel.

- Un coup « moyen » diminue SX de 10.
- Un coup intéressant (prise de pièce, échec) ne diminue SX que de 2 ou 3.
- Un coup peu intéressant diminue beaucoup SX (de l'ordre de 35).

Résultat : les coups intéressants seront explorés plus profondément.

Analyse des parties

Passer du temps là où il faut

Concentrer l'effort de recherche dans les *zones critiques*

- Si tous les débuts de partie se ressemblent, ne pas y consacrer trop de temps (ex : jouer au hasard pour le jeu des *amazones*)
- Passer plus de temps au milieu de partie (ex Awalé) car les fins de partie ne demandent pas beaucoup de temps (élagage très efficace dans l'Awalé).

Mais seule une connaissance profonde du jeu permet d'affiner les moments où la recherche de bon coups est la plus critique.

Analyse des parties

Passer du temps là où il faut

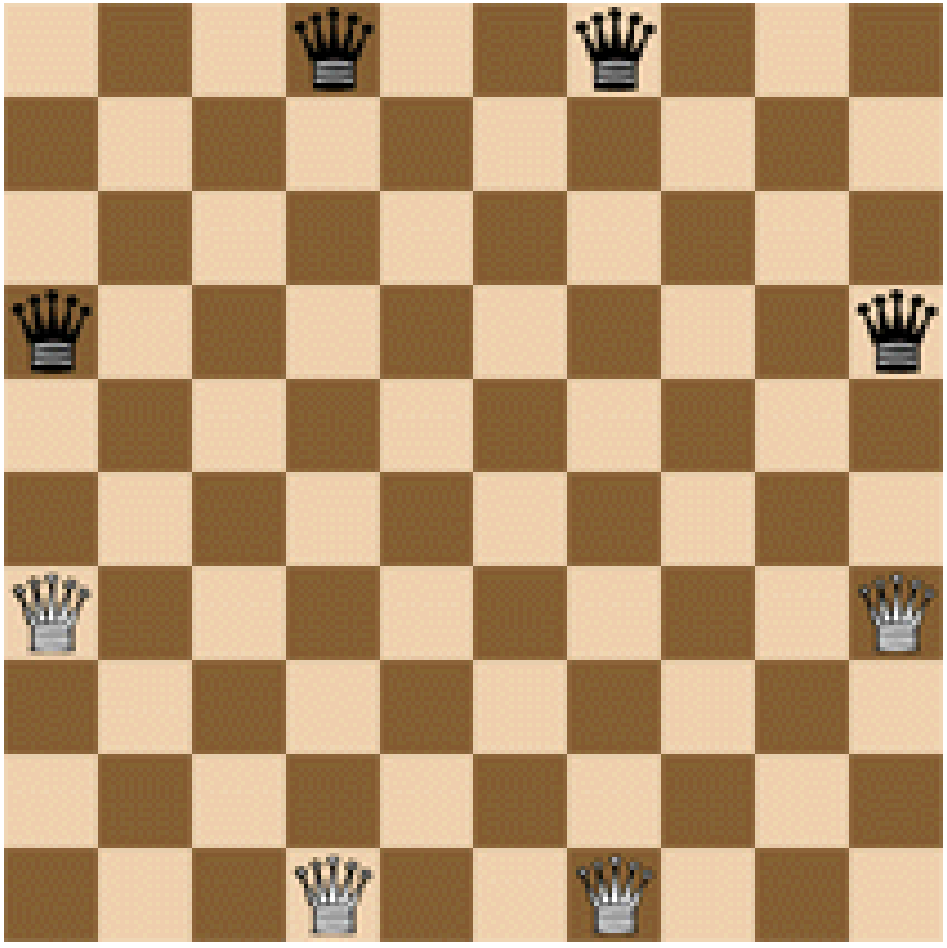
Concentrer l'effort de recherche dans les *zones critiques*

- Si tous les débuts de partie se ressemblent, ne pas y consacrer trop de temps (ex : jouer au hasard pour le jeu des *amazones*)
- Passer plus de temps au milieu de partie (ex Awalé) car les fins de partie ne demandent pas beaucoup de temps (élagage très efficace dans l'Awalé).

Mais seule une connaissance profonde du jeu permet d'affiner les moments où la recherche de bon coups est la plus critique.

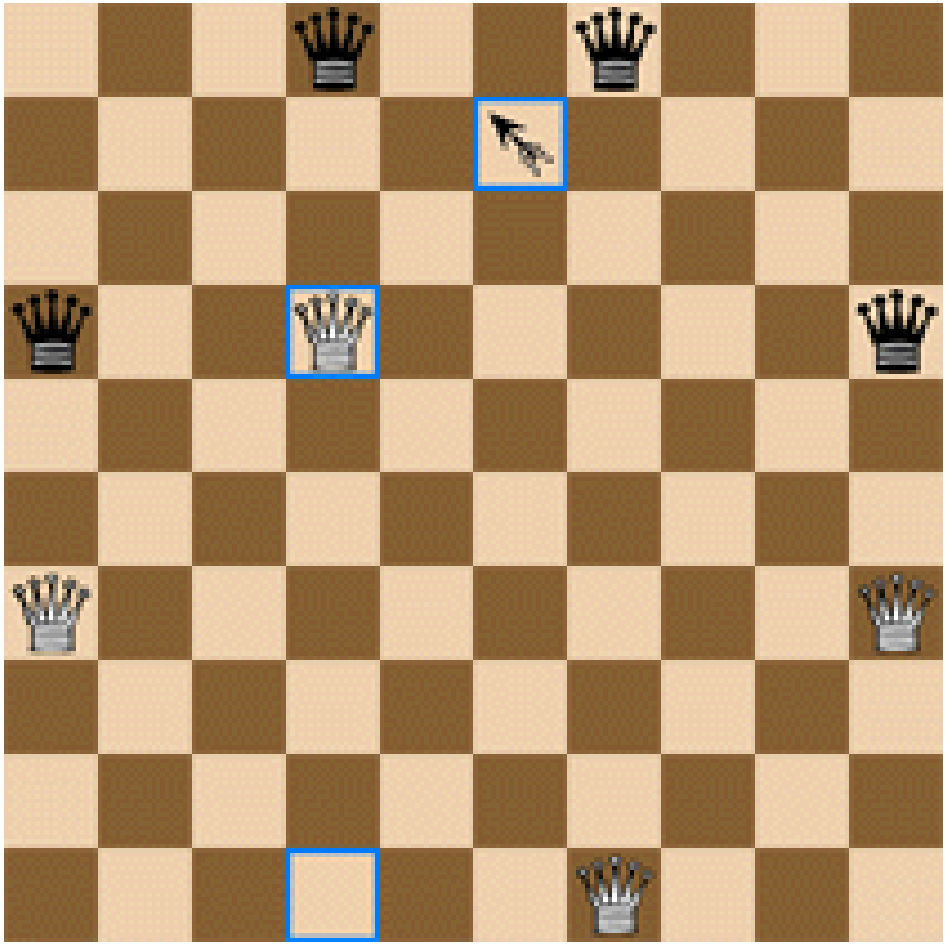
Exemple sur les Amazones

Comment gérer l'ouverture ?



Exemple sur les Amazones

Comment gérer l'ouverture ?



Analyse des parties

Passer du temps là où il faut

3 phases communes à tous les jeux

- Début de partie
- Milieu de partie
- Fin de partie

Un bon programme de jeu doit avoir un comportement fondamentalement différent dans chacune de ces trois phases.

Encore une heuristique

Il faut donc avoir une heuristique de haut niveau permettant de choisir entre les différentes heuristiques liées aux phases du jeu.

Analyse des parties

Passer du temps là où il faut

3 phases communes à tous les jeux

- Début de partie
- Milieu de partie
- Fin de partie

Un bon programme de jeu doit avoir un comportement fondamentalement différent dans chacune de ces trois phases.

Encore une heuristique

Il faut donc avoir une heuristique de haut niveau permettant de choisir entre les différentes heuristiques liées aux phases du jeu.

Agir sur les heuristiques

On peut faire évoluer la valeur de la fonction heuristique suivant l'avancement dans le jeu.

Exemples

- Awalé : faire des greniers en début de partie, puis simplement compter les graines en fin de partie
- Othello : Prendre des cases stratégiques, puis compter les pions.
- ...

La transition entre les différentes fonctions heuristiques doit être douce pour que l'horizon évolue doucement.

Début de partie : bibliothèques d'ouverture

Pour concevoir une bibliothèque, deux solutions :

- Utilisation d'un expert (recopier des ouvertures dans la littérature).
- Construction de ses propres bibliothèques. Utile lorsque le jeu n'a pas déjà été grandement étudié.
  - Lancement d'$\alpha\beta$ à grande profondeur (extension hors-ligne de l'horizon des premiers coups)
  - Faire jouer différentes ouvertures contre le programme lui-même

Comment stocker la bibliothèque ?

- Soit sous forme d'automate
- Soit sous forme de tables de transposition

Ne pas oublier d'introduire du hasard...

Début de partie : bibliothèques d'ouverture

Pour concevoir une bibliothèque, deux solutions :

- Utilisation d'un expert (recopier des ouvertures dans la littérature).
- Construction de ses propres bibliothèques. Utile lorsque le jeu n'a pas déjà été grandement étudié.
  - Lancement d'$\alpha\beta$ à grande profondeur (extension hors-ligne de l'horizon des premiers coups)
  - Faire jouer différentes ouvertures contre le programme lui-même

Comment stocker la bibliothèque ?

- Soit sous forme d'automate
- Soit sous forme de tables de transposition

Ne pas oublier d'introduire du hasard...

Début de partie : bibliothèques d'ouverture

Pour concevoir une bibliothèque, deux solutions :

- Utilisation d'un expert (recopier des ouvertures dans la littérature).
- Construction de ses propres bibliothèques. Utile lorsque le jeu n'a pas déjà été grandement étudié.
  - Lancement d'$\alpha\beta$ à grande profondeur (extension hors-ligne de l'horizon des premiers coups)
  - Faire jouer différentes ouvertures contre le programme lui-même

Comment stocker la bibliothèque ?

- Soit sous forme d'automate
- Soit sous forme de tables de transposition

Ne pas oublier d'introduire du hasard...

Début de partie : bibliothèques d'ouverture

Pour concevoir une bibliothèque, deux solutions :

- Utilisation d'un expert (recopier des ouvertures dans la littérature).
- Construction de ses propres bibliothèques. Utile lorsque le jeu n'a pas déjà été grandement étudié.
  - Lancement d'$\alpha\beta$ à grande profondeur (extension hors-ligne de l'horizon des premiers coups)
  - Faire jouer différentes ouvertures contre le programme lui-même

Comment stocker la bibliothèque ?

- Soit sous forme d'automate
- Soit sous forme de tables de transposition

Ne pas oublier d'introduire du hasard...

Début de partie : bibliothèques d'ouverture

Pour concevoir une bibliothèque, deux solutions :

- Utilisation d'un expert (recopier des ouvertures dans la littérature).
- Construction de ses propres bibliothèques. Utile lorsque le jeu n'a pas déjà été grandement étudié.
  - Lancement d'$\alpha\beta$ à grande profondeur (extension hors-ligne de l'horizon des premiers coups)
  - Faire jouer différentes ouvertures contre le programme lui-même

Comment stocker la bibliothèque ?

- Soit sous forme d'automate
- Soit sous forme de tables de transposition

Ne pas oublier d'introduire du hasard...

Milieu de partie

Domaine de $\alpha\beta$ avec tous les mécanismes vus :

- Recherche de coups profond
- Iterative Deepening
- Horizon de coups intéressants
- ...

Milieu de partie

Domaine de $\alpha\beta$ avec tous les mécanismes vus :

- Recherche de coups profond
- Iterative Deepening
- Horizon de coups intéressants
- ...

Fin de partie

Le domaine de la puissance brute

Généralement, le jeu est très contraint. Le facteur de branchement de l'arbre de jeu est réduit. On essaye d'aller jusqu'au bout de l'arbre de recherche pour trouver une stratégie gagnante.

- Les heuristiques sont généralement réduites à « gagné » « perdu ».
- Il faut penser à augmenter considérablement l'horizon.

Question

Peut-on décider si on a gagné ou perdu sans aller au bout de l'arbre ?

Fin de partie

Le domaine de la puissance brute

Généralement, le jeu est très contraint. Le facteur de branchement de l'arbre de jeu est réduit. On essaye d'aller jusqu'au bout de l'arbre de recherche pour trouver une stratégie gagnante.

- Les heuristiques sont généralement réduites à « gagné » « perdu ».
- Il faut penser à augmenter considérablement l'horizon.

Question

Peut-on décider si on a gagné ou perdu sans aller au bout de l'arbre ?

Fin de partie

Le domaine de la puissance brute

Généralement, le jeu est très contraint. Le facteur de branchement de l'arbre de jeu est réduit. On essaye d'aller jusqu'au bout de l'arbre de recherche pour trouver une stratégie gagnante.

- Les heuristiques sont généralement réduites à « gagné » « perdu ».
- Il faut penser à augmenter considérablement l'horizon.

Question

Peut-on décider si on a gagné ou perdu sans aller au bout de l'arbre ?

Fin de partie

Le domaine de la puissance brute

Généralement, le jeu est très contraint. Le facteur de branchement de l'arbre de jeu est réduit. On essaye d'aller jusqu'au bout de l'arbre de recherche pour trouver une stratégie gagnante.

- Les heuristiques sont généralement réduites à « gagné » « perdu ».
- Il faut penser à augmenter considérablement l'horizon.

Question

Peut-on décider si on a gagné ou perdu sans aller au bout de l'arbre ?

Fin de partie

Le domaine de la puissance brute

Généralement, le jeu est très contraint. Le facteur de branchement de l'arbre de jeu est réduit. On essaye d'aller jusqu'au bout de l'arbre de recherche pour trouver une stratégie gagnante.

- Les heuristiques sont généralement réduites à « gagné » « perdu ».
- Il faut penser à augmenter considérablement l'horizon.

Question

Peut-on décider si on a gagné ou perdu sans aller au bout de l'arbre ?

Petite conclusion

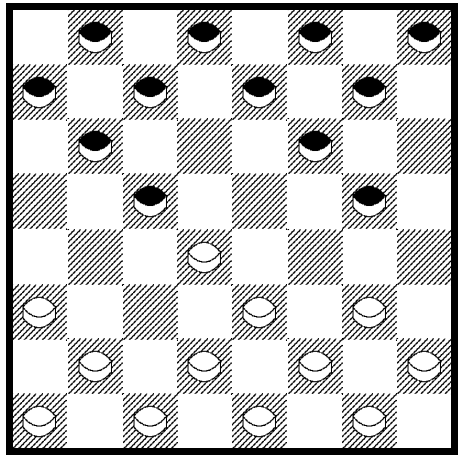
Synthèse expérimentale

Sur l'Othello

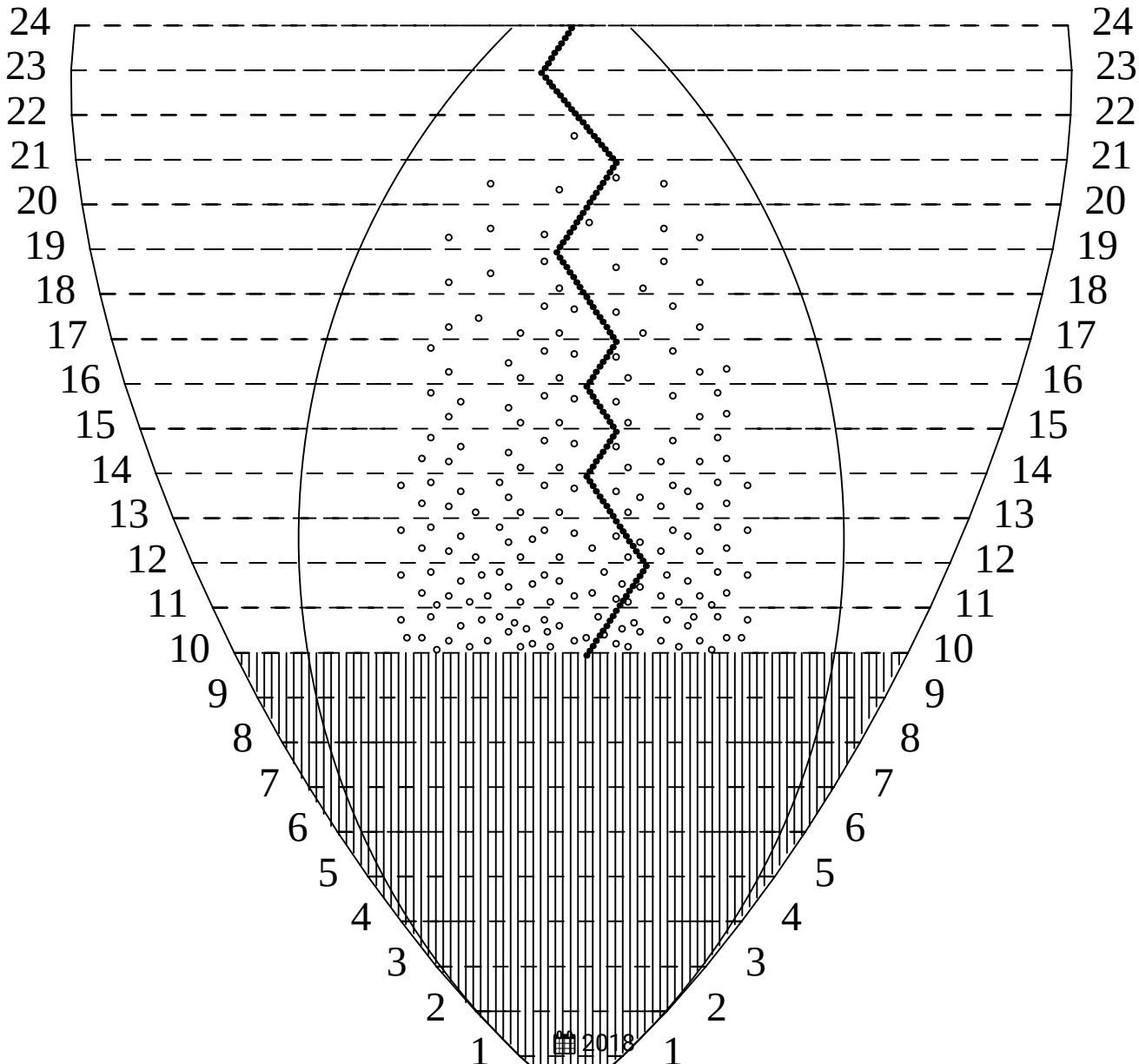
p	minimax	Scout	α-β	SSS*
1	3 – 0,37	4 – 0,45	3 – 0,36	3 – 0,36
2	14 – 1,38	21 – 2.09	13 – 1,4	11 – 1,19
3	61 – 6,0	45 – 5,0	37 – 3,9	35 – 3,7
4	349 – 32,5	246 – 23,7	150 – 14,77	95 – 10,0
5	2050 – 185,7	615 – 61,6	418 – 41,4	292 – 19,2
6	13773 – 1213,5	2680 – 254,5	1830 – 172,9	1617 – 117,3

Fermeture des Dames

L'un des prix IJCAI'05 a été donné à Schaeffer pour avoir notamment prouvé que l'ouverture aux dames ci-dessous n'était pas gagnante ni perdante :



Fermeture des Dames



Dans les profondeurs de *Deep Blue*

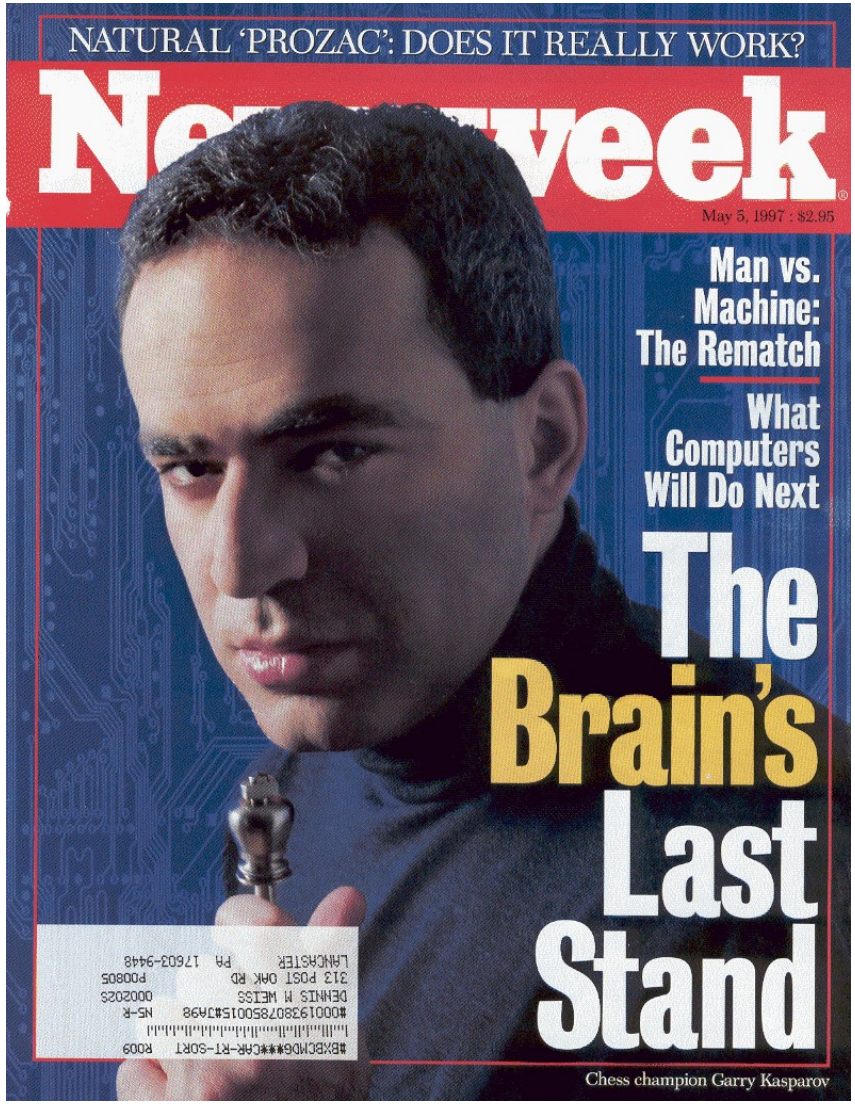
1996 : Kasparov gagne

Kasparov 4 / Deep Blue 2
(2 Victoires / 4 Nuls)

- “It was a wonderful and extremely human move,”
- “I had played a lot of computers but had never experienced anything like this. I could feel — I could smell — a new kind of intelligence across the table.”

Kasparov exprime son étonnement par rapport au niveau de jeu de Deep Blue

1997 : Le rematch



Kasparov sait que Deep Blue a été grandement amélioré

Puissance de Deeper Blue

Version de 1997

- 200 Millions de positions par seconde (évaluation)
- 11 GFlops (259ieme super ordinateur)
- 6-8 coups de profondeur
- Jusqu'à 20 coups analysés par plateau

Jusqu'à deux fois plus puissante que la version de 1996

Gros enjeux !

Time

WHEN GARRY KASPAROV FACED OFF AGAINST AN IBM COMPUTER in last month's celebrated chess match, he wasn't just after more fame and money. (...) the world chess champion was playing for you, me, the whole human species. He was trying, as he put it shortly before the match, to "help defend our dignity."

Kasparov vs Deep(er) Blue

Victoire de Deep(er) Blue : 3.5 / 2.5

1er match

Deep Blue perd mais fait un coup ”surhumain”. Sans gain immédiat. Vision à long terme du jeu ?

2nd match

Kasparov abandonne brutalement (même si un nul semblait possible)

D’après le Time (`http://time.com/3705316/deep-blue-kasparov/`) :
”He was again riled by a move the computer made that was so surprising, so un-machine-like, that he was sure the IBM team had cheated”

Kasparov vs Deep(er) Blue

Victoire de Deep(er) Blue : 3.5 / 2.5

1er match

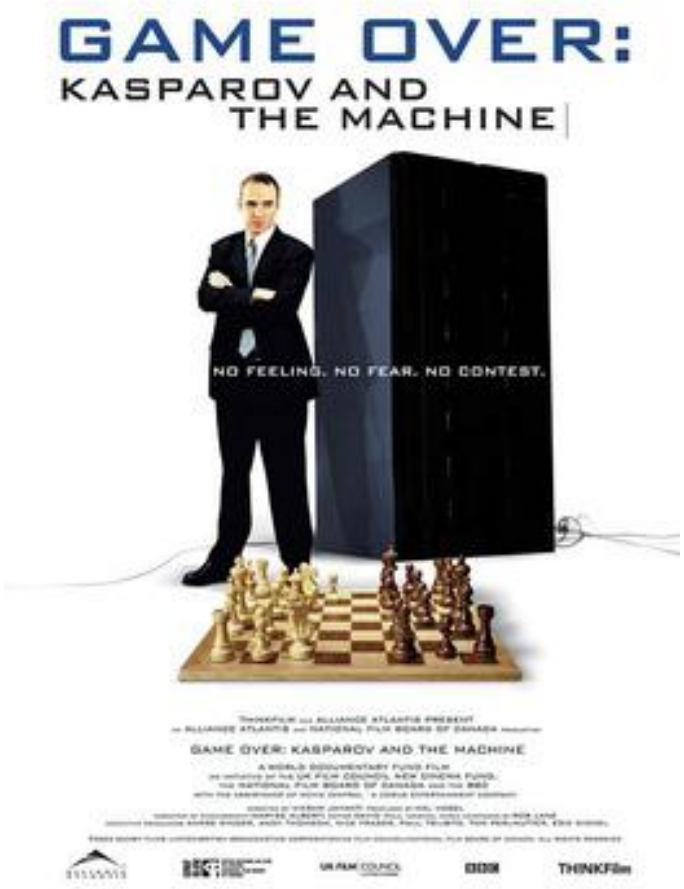
Deep Blue perd mais fait un coup ”surhumain”. Sans gain immédiat. Vision à long terme du jeu ?

2nd match

Kasparov abandonne brutalement (même si un nul semblait possible)

D’après le Time (`http://time.com/3705316/deep-blue-kasparov/`) :
”He was again riled by a move the computer made that was so surprising, so un-machine-like, that he was sure the IBM team had cheated”

Kasparov ne comprend pas



- Kasparov demande l'accès aux logs
- L'équipe d'IBM est "ventilée"
- Les conditions du match sont discutées

Deep(er) Blue : le coup ”surhumain”

Deep(er) Blue a gagné psychologiquement sur son 44ième coup, 1ère manche.

Comment les ingénieurs d’IBM ont réussi cela ?

- Coup à très long terme
- Temps de réflexion (anormalement) long

La réponse a été donnée dans le Washington Post

Deep(er) Blue : le coup ”surhumain”

C’était un bug

Aujourd'hui les échecs

Alphazero a battu Stockfish (100-0)

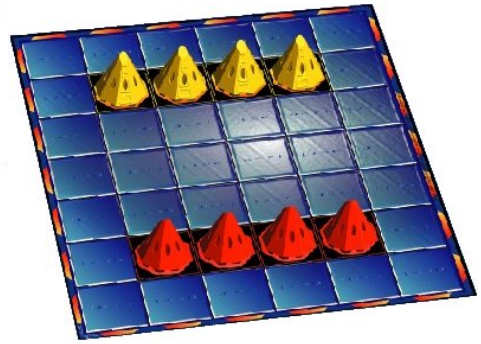
en 2006 : Le champion du monde a perdu (4-2) contre Deep Fritz

- sur un dual core Xeon
- 8 Millions de positions par secondes
- profondeur de 17-18 (heuristiques).

OCTI

Un jeu de 2002 créé pour résister aux machines

À vos idées !



Infos sur `www.octi.net`

Ce qui était vrai en 1960...

En guise de conclusion

[Samuel, 1960] « Just as it was impossible to begin the discussion of game-playing machines without referring to the hoaxes of the past, it is equally unthinkable to close the discussion without a diagnosis. Programming computers to play games is but one stage in the development of an understanding of the methods which must be employed for the machine simulation of intellectual behavior. As we progress in this understanding it seems reasonable to assume that these newer techniques will be applied to real-life situations with increasing frequency, and the effort devoted to games... will decrease. Perhaps we have not yet reached this turning point, and we may still have much to learn the study of games. »

Cette citation est toujours d'actualité... 45 ans après...