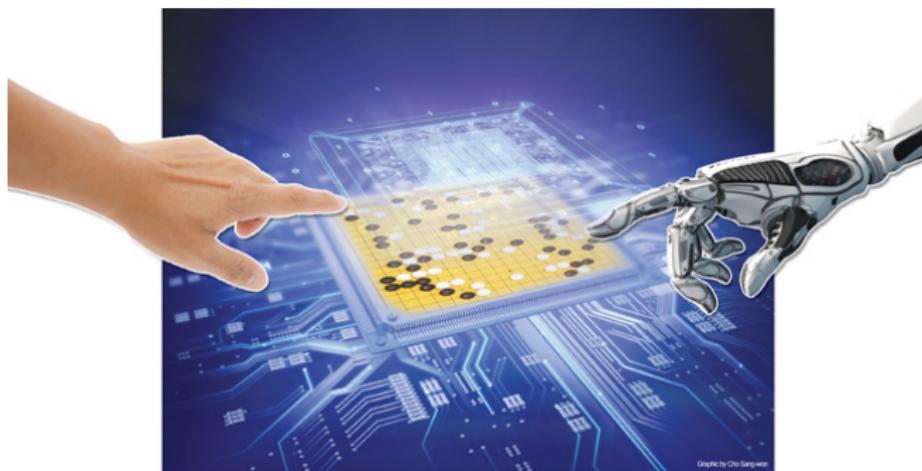




# Jeu de Go et Exploration d'Arbre par Bandit

CentraleSupélec – Gif

## IA et Jeu de Go



- 2016 : AlphaGo bat le meilleur joueur humain
- Combine des méthodes de **deep learning** avec une exploration d'arbre par bandit



## Plan

- 1 IA et Jeu de Go
- 2 Avant l'Exploration d'Arbre par Bandit
- 3 Exploration d'Arbre par Bandit
- 4 Conclusion

## Pourquoi une IA pour le jeu de go ?



- un jeu de plateau qui a longtemps résisté aux IA
- règles **simples**
- méthodes classiques (alphabeta) inefficaces

## Histoire



- aurait été inventé en Chine en 2000 BC
- premiers écrits : 500 BC
- fait parti des 4 arts majeurs chinois : peinture, calligraphie, guqin, go
- se répand en Asie dès 800 dans la noblesse
- aujourd'hui, environ 20 millions de joueurs

*If aliens exist they might play chess, but they certainly play Go.*

Edward Lasker

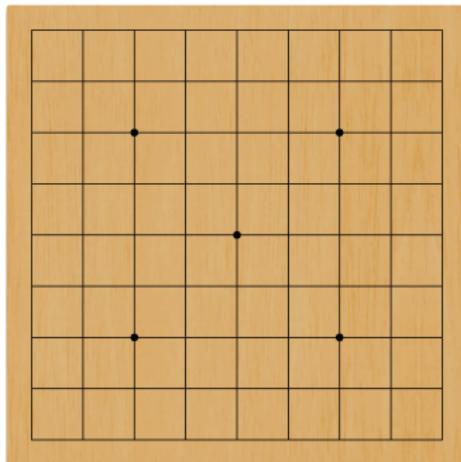
## Matériel

- plateau de jeu : Goban
- deux tailles 9x9 ou 19x19
- pierres noires et blanches



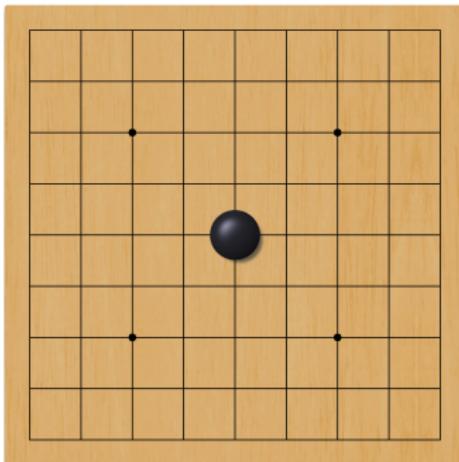
## Règles : placement

- Début de la partie : le plateau est vide
- Chaque joueur pose une pierre à tour de rôle
- Noir commence
- Pierres posées sur les intersections



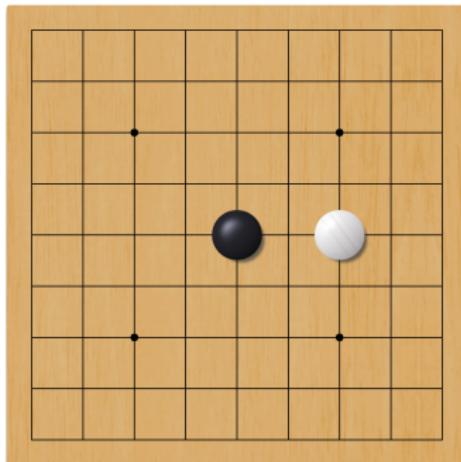
## Règles : placement

- Début de la partie : le plateau est vide
- Chaque joueur pose une pierre à tour de rôle
- Noir commence
- Pierres posées sur les intersections



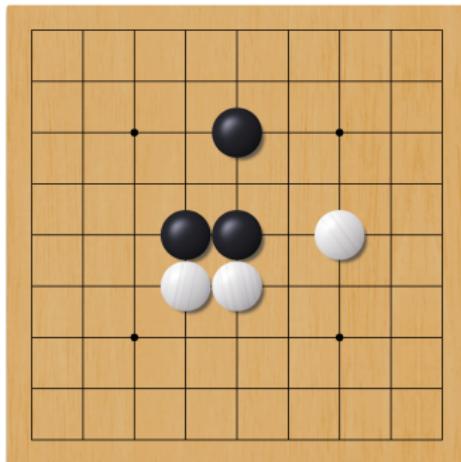
## Règles : placement

- Début de la partie : le plateau est vide
- Chaque joueur pose une pierre à tour de rôle
- Noir commence
- Pierres posées sur les intersections



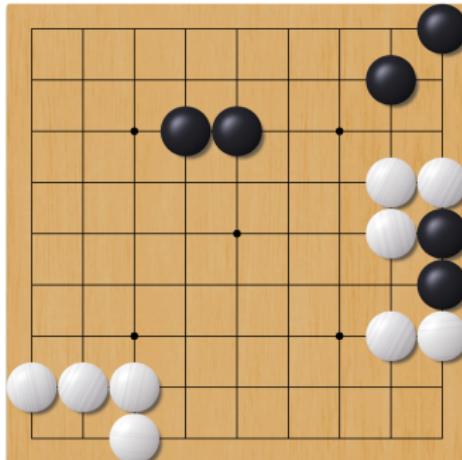
## Règles : placement

- Début de la partie : le plateau est vide
- Chaque joueur pose une pierre à tour de rôle
- Noir commence
- Pierres posées sur les intersections



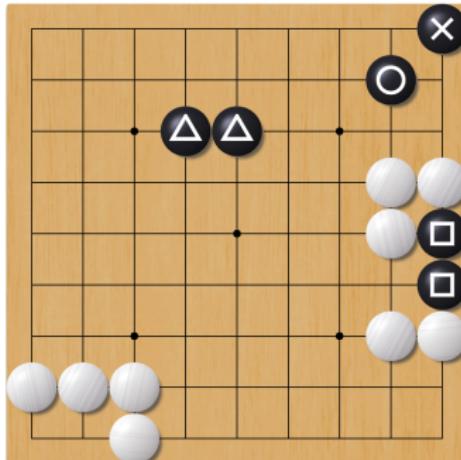
## Règles : chaînes et libertés

- Pierres reliées horizontalement ou verticalement : **une chaîne**
- Emplacements libres autour d'une chaîne : **liberté**



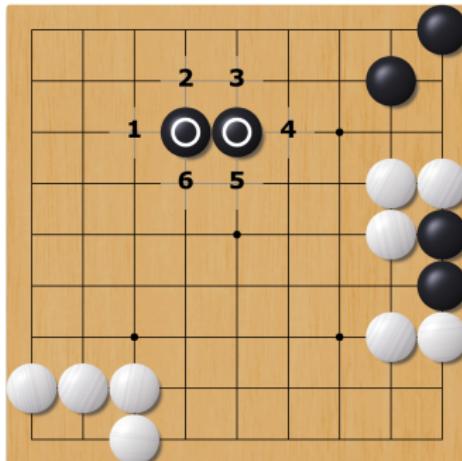
## Règles : chaînes et libertés

- Pierres reliées horizontalement ou verticalement : **une chaîne**
- Emplacements libres autour d'une chaîne : **liberté**



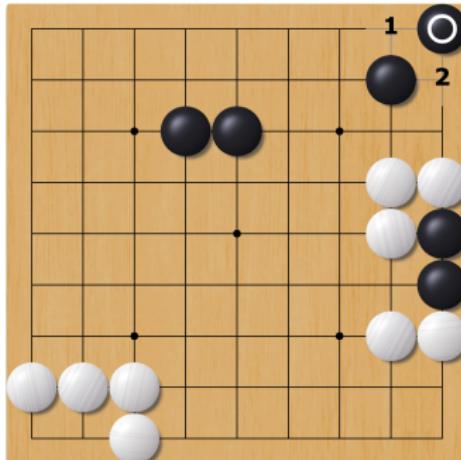
## Règles : chaînes et libertés

- Pierres reliées horizontalement ou verticalement : **une chaîne**
- Emplacements libres autour d'une chaîne : **liberté**



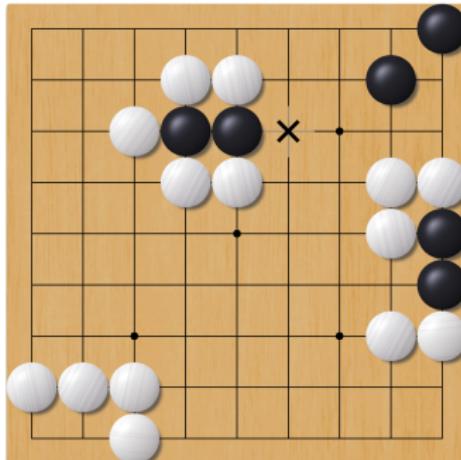
## Règles : chaînes et libertés

- Pierres reliées horizontalement ou verticalement : **une chaîne**
- Emplacements libres autour d'une chaîne : **liberté**



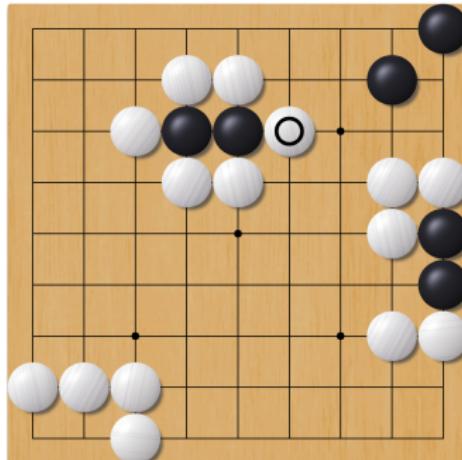
## Règles : capture

- Enlever la dernière liberté d'une chaîne : **capture**
- les pierres sont enlevées du plateau



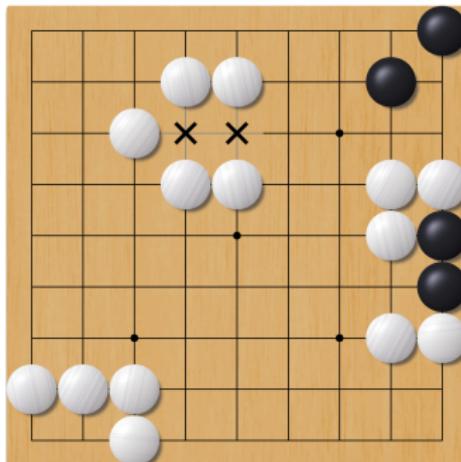
## Règles : capture

- Enlever la dernière liberté d'une chaîne : **capture**
- les pierres sont enlevées du plateau



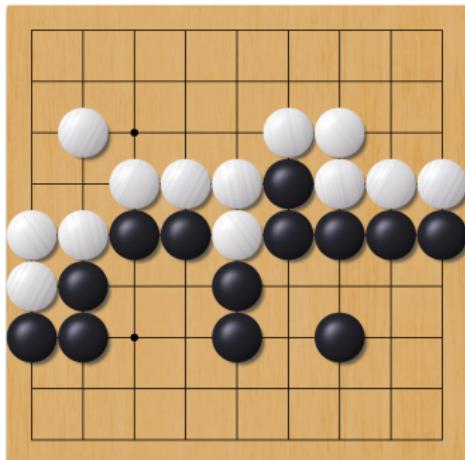
## Règles : capture

- Enlever la dernière liberté d'une chaîne : **capture**
- les pierres sont enlevées du plateau



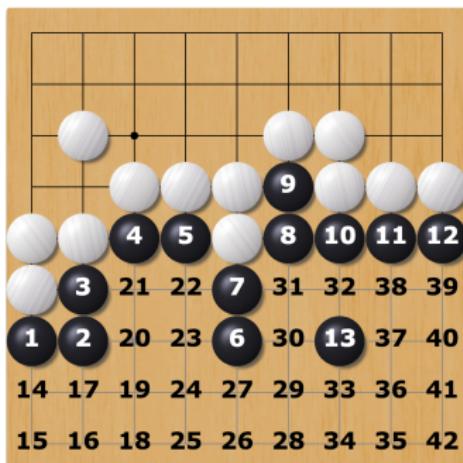
## Règles : fin de partie

- partie terminée quand les deux joueurs passent
- score : nb de pierres + territoire + bonus pour blanc (komi)



## Règles : fin de partie

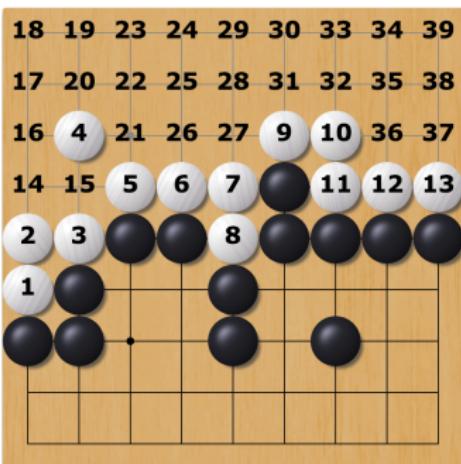
- partie terminée quand les deux joueurs passent
- score : nb de pierres + territoire + bonus pour blanc (komi)



score noir : 13 (pierres) + 29 (territoire)=**42**

## Règles : fin de partie

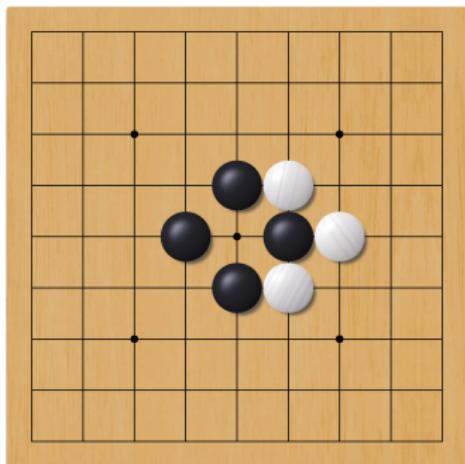
- partie terminée quand les deux joueurs passent
- score : nb de pierres + territoire + bonus pour blanc (komi)



score blanc : 13 (pierres) + 26 (territoire) + 7.5 (komi)=**46.5**

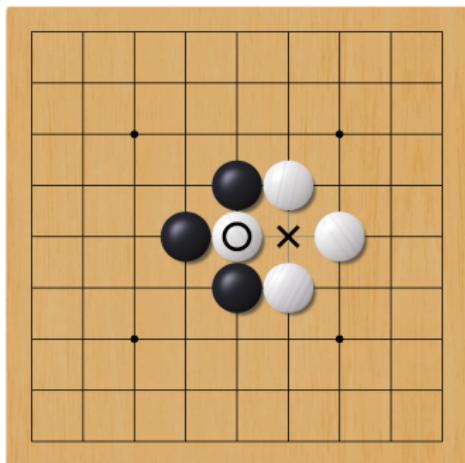


## Règles : le ko



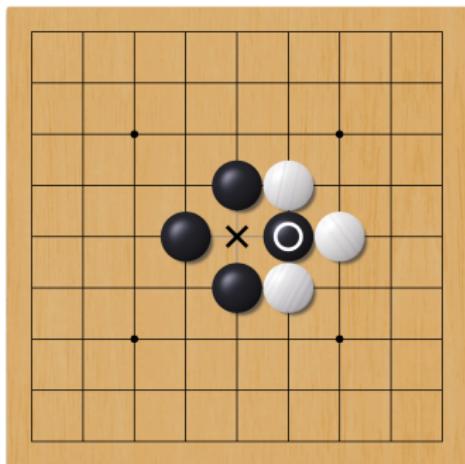


## Règles : le ko

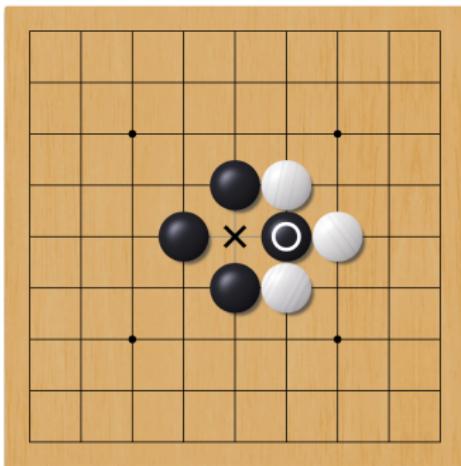




## Règles : le ko

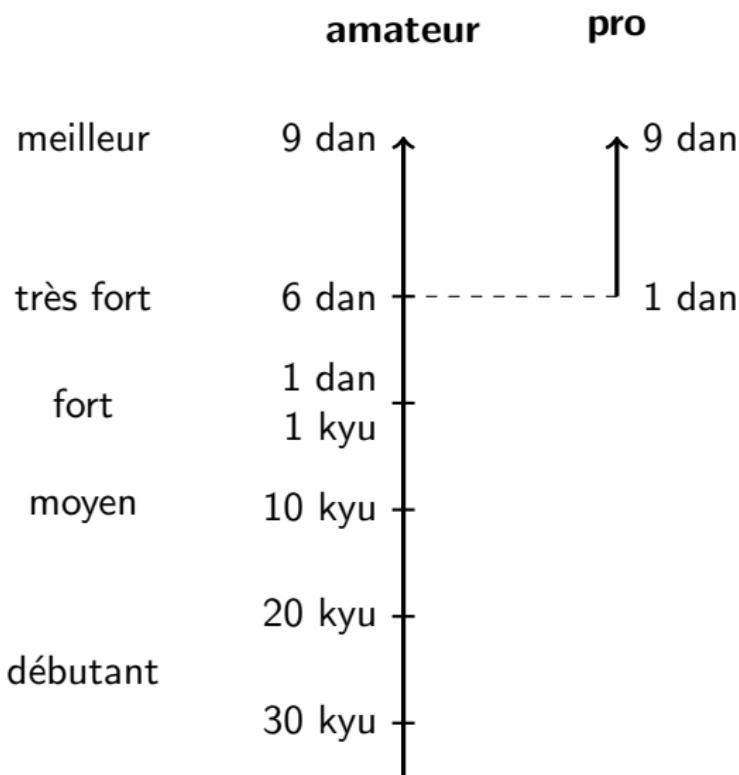


## Règles : le ko



- problème : captures répétées successives
- règle (humain) : pas le droit de remettre le plateau dans l'état juste avant
- règle (ordinateur) : pas le droit de remettre le plateau dans n'importe quel état précédent

## Échelle de niveau



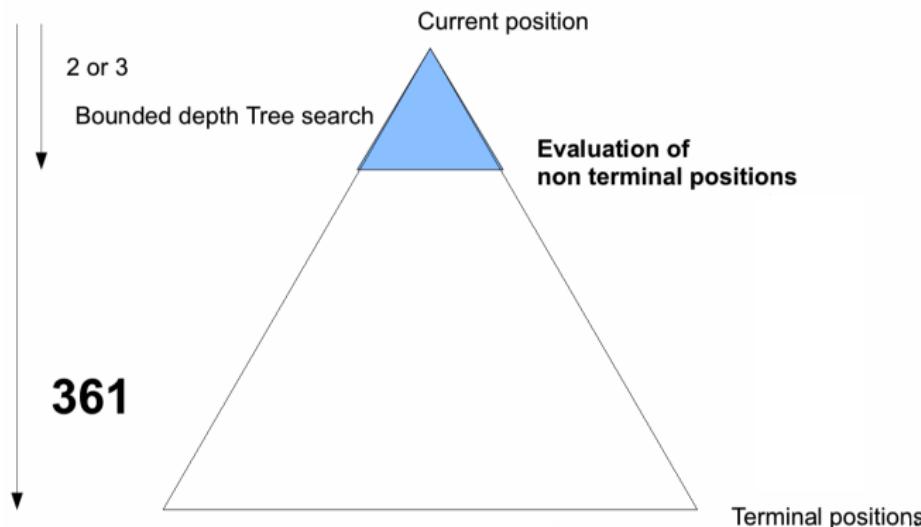


## Plan

- 1 IA et Jeu de Go
- 2 Avant l'Exploration d'Arbre par Bandit
- 3 Exploration d'Arbre par Bandit
- 4 Conclusion

## Principe

- Exploration d'arbre alphabeta
- Évaluation des nœuds basée sur des connaissances expertes

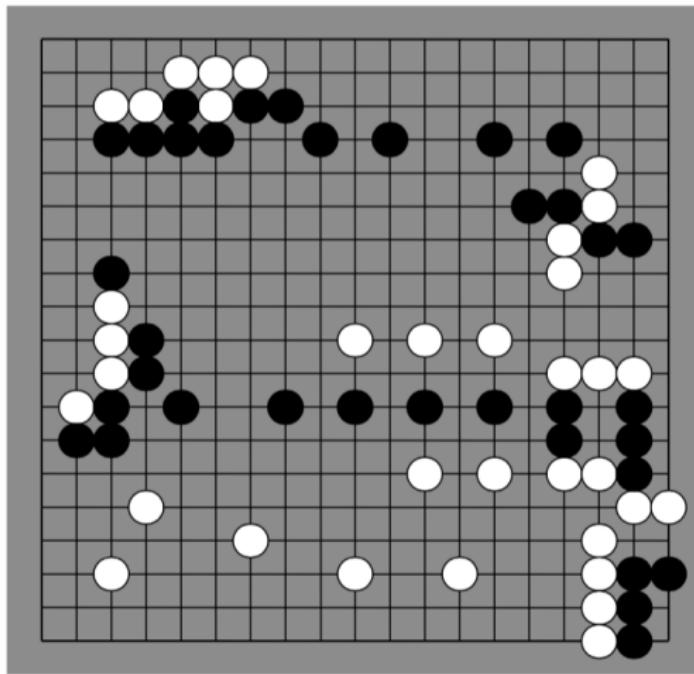


## Évaluation

- Découpage du plateau en sous parties
- Évaluation de chaque sous partie par **recherche locale**  
(souvent alphabeta)
- groupe mort, vivant, territoire, ...
- Recomposition d'un score global

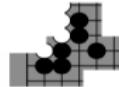
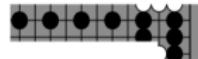
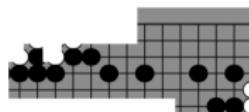


## Position à évaluer



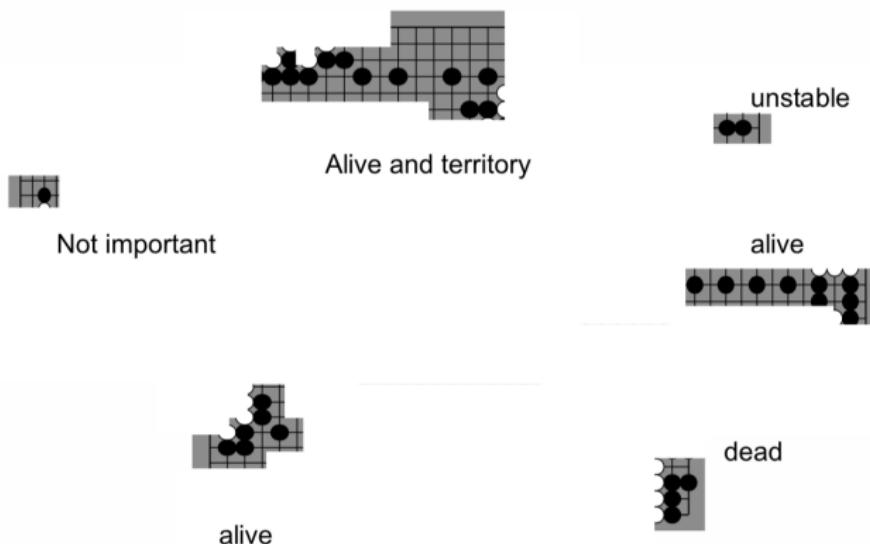


## Découpage du plateau



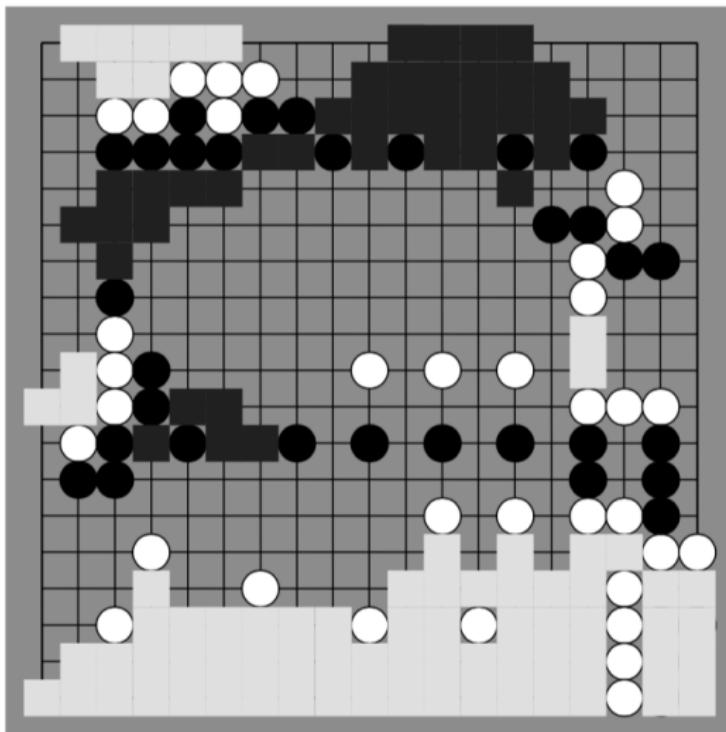


## Évaluation locale





## Évaluation globale





## Avantages

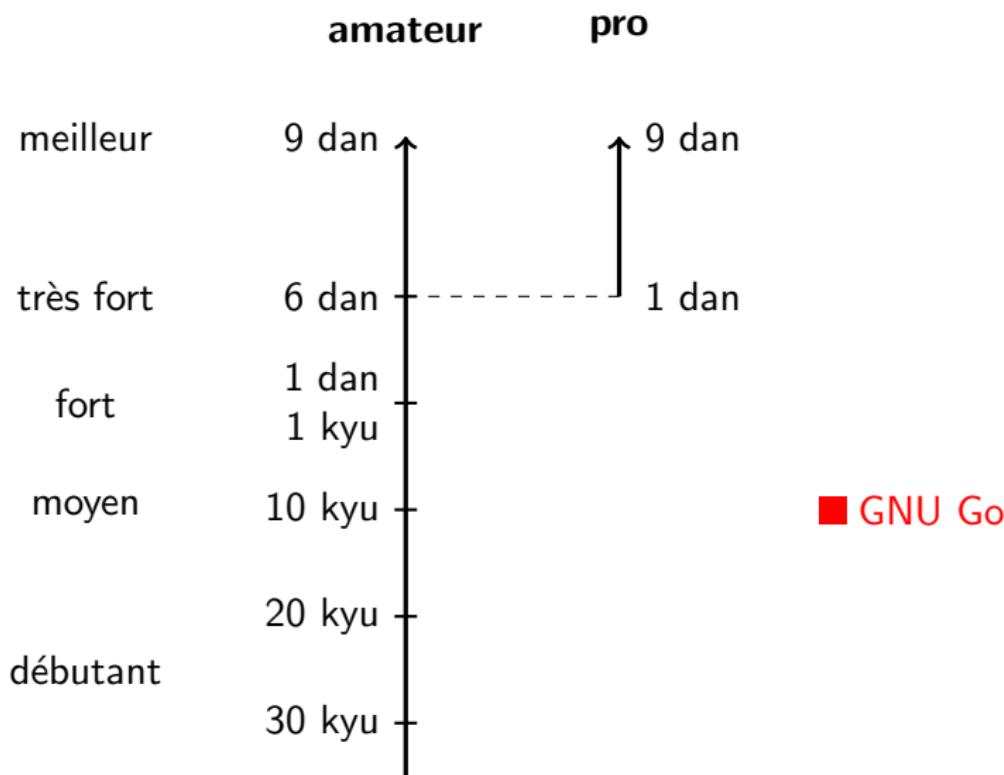
- Algorithme très rapide
- Évaluation locale peut être très performante



## Inconvénients

- Découpage et recomposition difficile et ayant un fort impact
- Pas d'**interaction** entre les positions locales
- Demande beaucoup de **connaissances expertes**

## Échelle de niveau



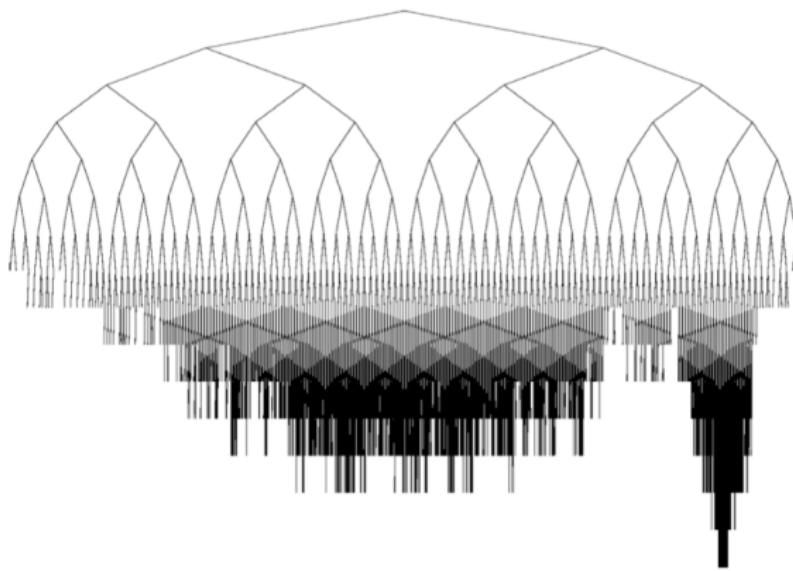


## Plan

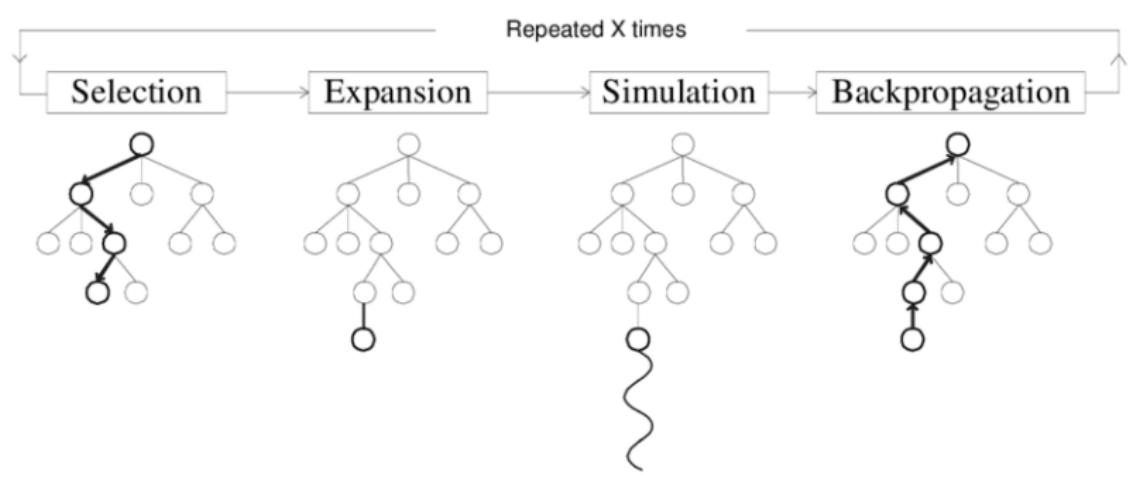
- 1 IA et Jeu de Go
- 2 Avant l'Exploration d'Arbre par Bandit
- 3 Exploration d'Arbre par Bandit
  - Construction de l'Arbre
  - Problème de Bandit
- 4 Conclusion

## Idée

- Certains coups sont mauvais, inutile d'explorer leurs enfants
- construction d'un arbre **déséquilibré**
- branches plus intéressantes explorées plus profondément
- Construction **itérative**

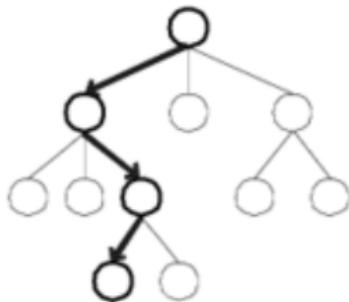


## Principe



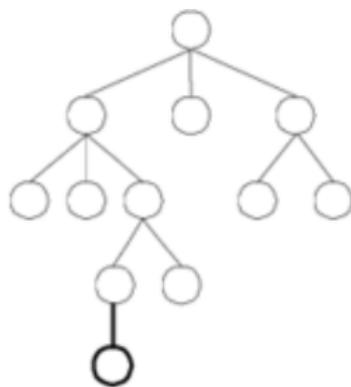
[Chaslot et al., 2008]

## Descente



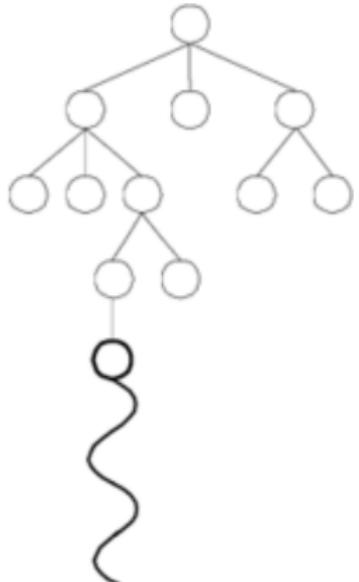
- partir de la **racine**
- **choisir** un nœud fils (détailé plus tard)
- **répéter** récursivement
- s'arrêter dès qu'on atteint un nœud en dehors de l'arbre (le nœud *frontier*)

## Extension



- ajouter le nœud *frontier* à l'arbre

## Evaluation



### Monte Carlo

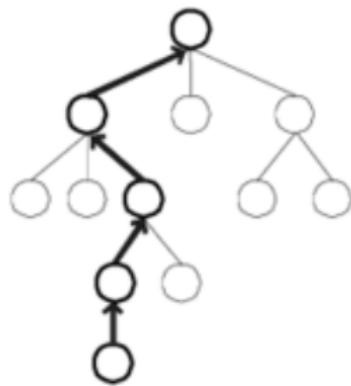
- Choix **aléatoire** d'un coup jusqu'à la fin de la partie

### Propriétés

- méthode générique
- facile à implémenter
- **non biaisé**
- possibilité d'intégrer des connaissances expertes en utilisant une distribution non uniforme

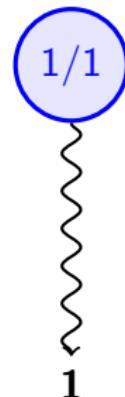
A l'origine du nom **Monte Carlo Tree Search**

## Backpropagation

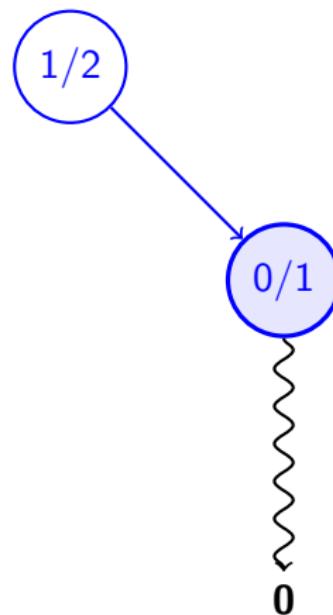


- appliquée aux nœuds de l'arbre visités pendant la descente
- **mise à jour des valeurs** en fonctions de l'évaluation

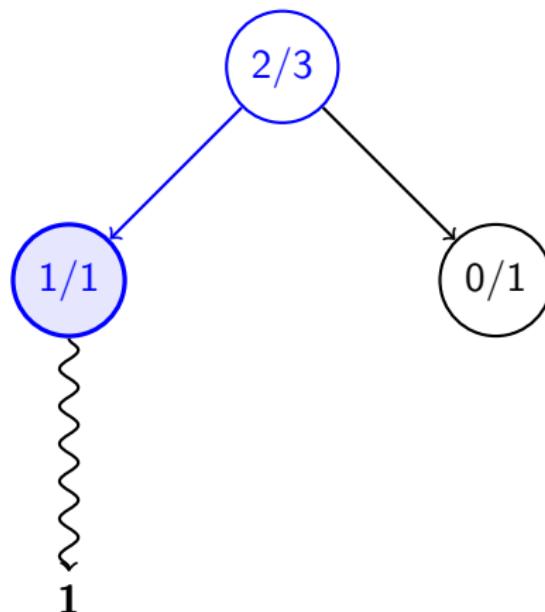
## Exemple



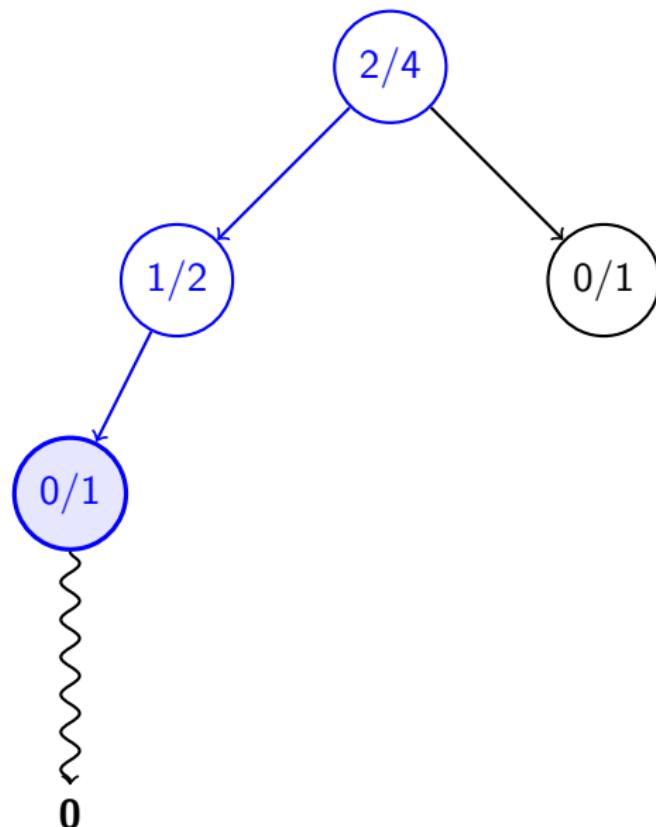
## Exemple



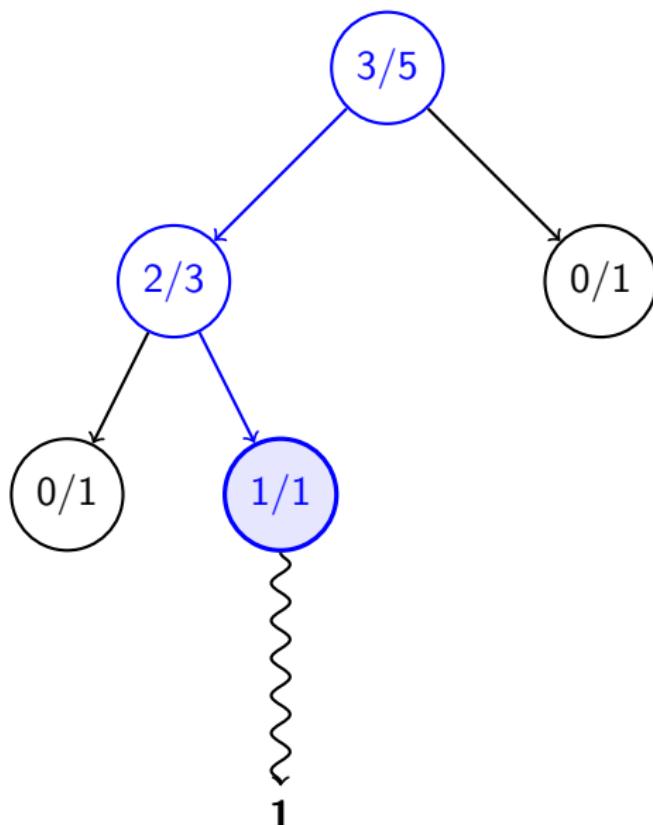
## Exemple



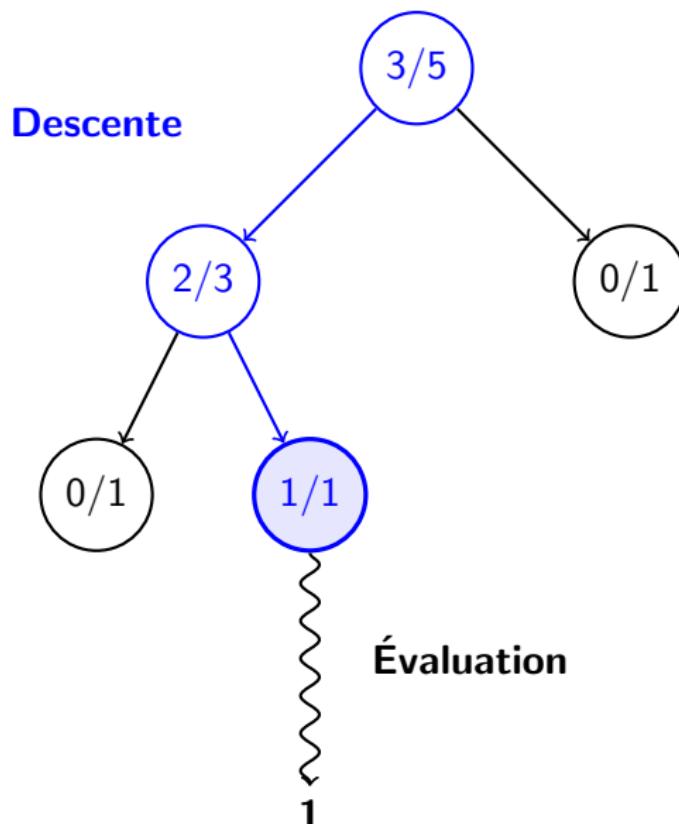
## Exemple



## Exemple



## Exemple



## Introduction du problème



Dans un casino, il y a plusieurs machines à sous différentes en terme de récompense.

- Comment répartir mes pièces entre les machines ?

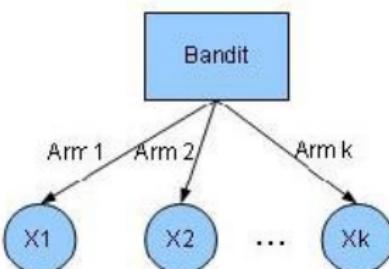
## Autres problèmes similaires



- *Essais cliniques* : trouver le traitement qui fonctionne le mieux.
- *Sélection d'un serveur dans un réseau* : trouver le serveur avec le temps de réponse le plus faible.
- *Publicité ciblée* : trouver le type de pub qui intéressera le plus un utilisateur.
- ...

Ce sont des problèmes où on a **plusieurs fois le même choix** à effectuer. Le choix conduit à une **récompense aléatoire**.

## Définition formelle



- un ensemble de bras  $A = \{1, \dots, K\}$ .
- chaque bras est associé à une distribution de probabilité  $X_k$  d'espérance  $\mu_k$ .
- l'algorithme choisit un bras  $a$  à chaque pas de temps.
- le bandit retourne une récompense  $r$  : une réalisation de  $X_a$ .
- les tirages successifs sur un même bras sont indépendant et identiquement distribués.



## Notations supplémentaires

- $T_i(n)$  : le nombre de fois que le bras  $i$  a été sélectionné au pas de temps  $n$ .
- $\mu^* = \max_{1 \leq i \leq K} \mu_i$
- $\Delta_i = \mu^* - \mu_i$
- $\Delta = \min_{i: \Delta_i > 0} \Delta_i$



## Objectif

Le but est d'optimiser le **regret**  $R_n$  défini comme suit :

$$R_n = \mu^* n - \mathbb{E} \sum_{j=1}^K T_j(n) \mu_j$$

$$R_n = \sum_{j=1}^K \Delta_j \mathbb{E}[T_j(n)]$$



## Borne inférieure

Pour toute stratégie d'allocation et pour tout bras non optimal :

$$\mathbb{E}[T_j(n)] \geq C * \log n$$

où  $C$  est une constante.

On en déduit que le meilleur regret atteignable est en  $\mathcal{O}(\log(n))$ .

[Lai and Robbins, 1985]



## UCB

Principe de l'algorithme :

- A partir des informations disponibles au temps  $t$ , on calcule la borne de confiance supérieur (UCB) correspondant à chaque bras.
- On choisit le bras qui a la valeur UCB la plus grande.

[Auer et al., 2002]



## UCB

Calcul de la valeur UCB pour le bras  $i$  au pas de temps  $t$  :

$$\hat{\mu}_{i,t-1} + \sqrt{\frac{3 \log(t)}{2 T_i(t-1)}}$$

où  $\hat{\mu}_{i,t-1}$  correspond à la moyenne empirique du bras  $i$ .



## UCB

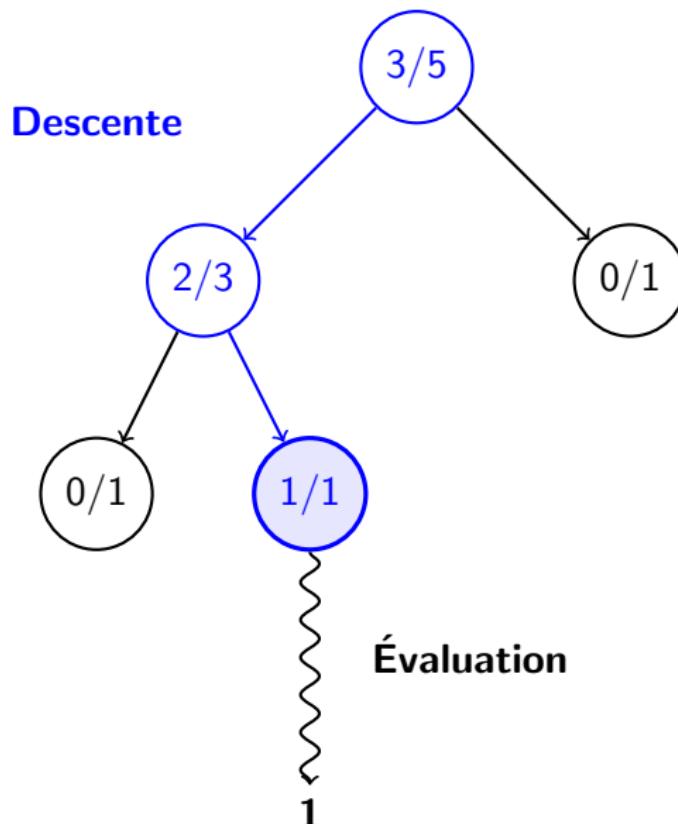
Borne sur le regret :

$$R_n \leq 6 * \sum_{i \neq i^*} \frac{\log(n)}{\Delta_i} + C$$

où  $C$  est une constante.

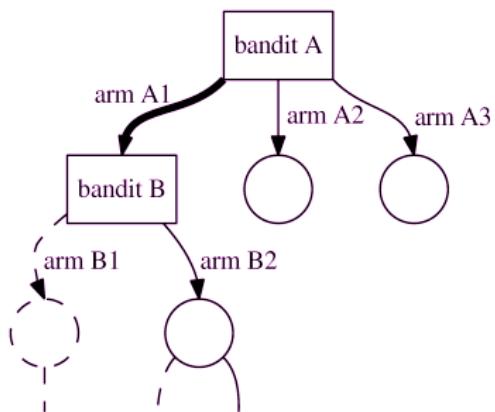
On en déduit que UCB est asymptotiquement optimal.

## Rappel



## Descente dans l'arbre

La descente dans l'arbre se fait en considérant que chaque choix d'une branche est un problème de bandit.



On utilise UCB pour le résoudre.

MCTS + UCB = UCT (UCB applied to Tree)

[Kocsis and Szepesvári, 2006]



## UCT en pratique

- Ajout d'un paramètre  $p$  de contrôle de l'exploration :

$$\hat{\mu}_{i,t-1} + p \sqrt{\frac{\log(t)}{T_i(t-1)}}$$

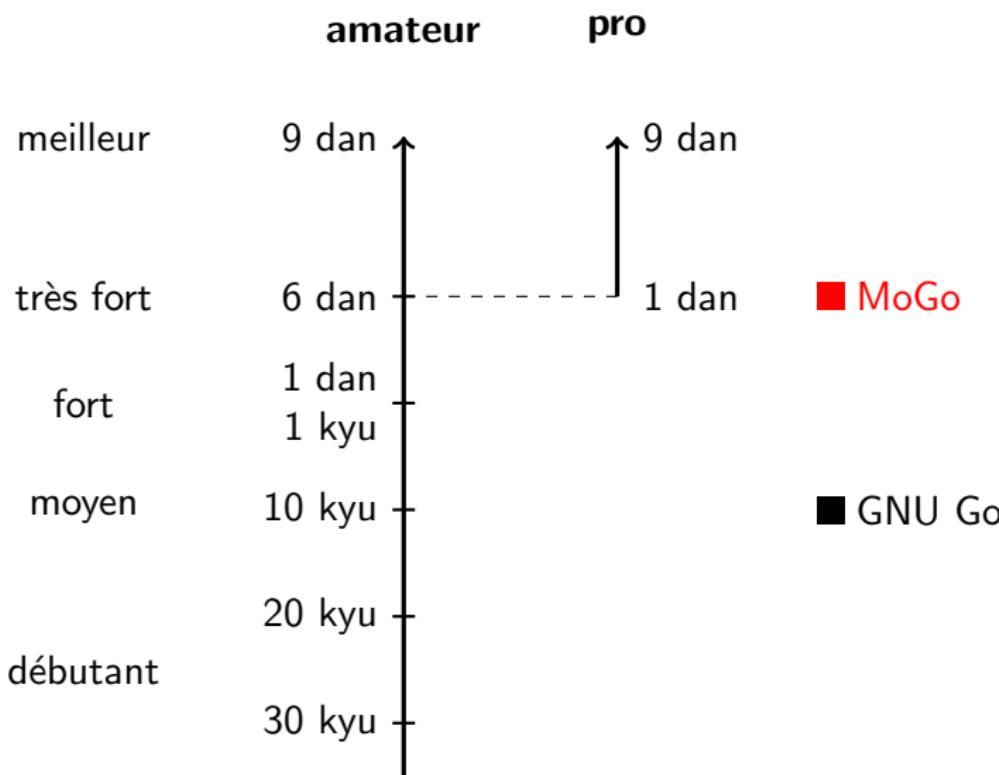
- Ajout de connaissances a priori  $C_i(t)$  :

$$\hat{\mu}_{i,t-1} + p \sqrt{\frac{\log(t)}{T_i(t-1)}} + C_i(t)$$

## Améliorations

- *Ordonner les bras du bandit*
- limiter le nombre de bras explorés
- *Ajouter de connaissances expertes* [Chaslot et al., 2009]
- modifier la formule de bandit ou la distribution de probabilité pour l'évaluation
- *Rapide Action Value Estimation (RAVE)*  
[Gelly and Silver, 2007]
- prendre en compte le fait qu'inverser la séquence de coup peut donner le même résultat
- ...

## Échelle de niveau



## Première victoire en 9x9 en tant que joueur noir

- programme : MoGoTW
- MCTS avec de nombreuses améliorations
- 32 machines de 8 cœurs
- adversaire : C.-H. Chou (pro 9 dan)



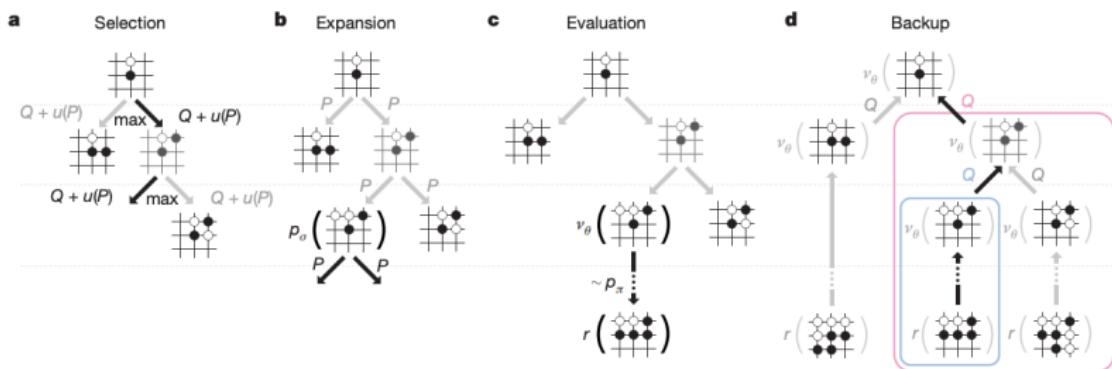
[Lee et al., 2009]



## Plan

- 1 IA et Jeu de Go
- 2 Avant l'Exploration d'Arbre par Bandit
- 3 Exploration d'Arbre par Bandit
- 4 Conclusion

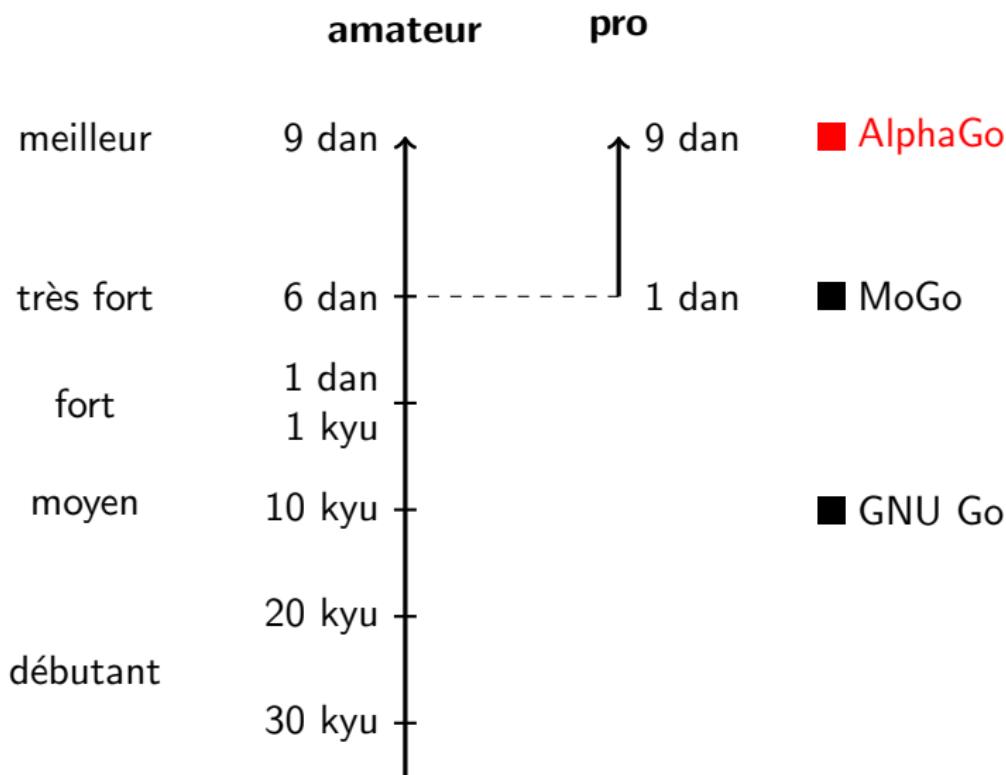
# AlphaGo



- Utilisation de 3 réseaux de neurones profonds
- guider la **descente** dans l'arbre ( $p_\sigma$ )
- choisir les coups lors de l'**évaluation** ( $p_\pi$ )
- prédire la **valeur** d'un nœud ( $v_\theta$ )
- le score propagé dépend de l'évaluation et de la valeur prédictive

[Silver et al., 2016]

## Échelle de niveau



## Exemple d'autres applications

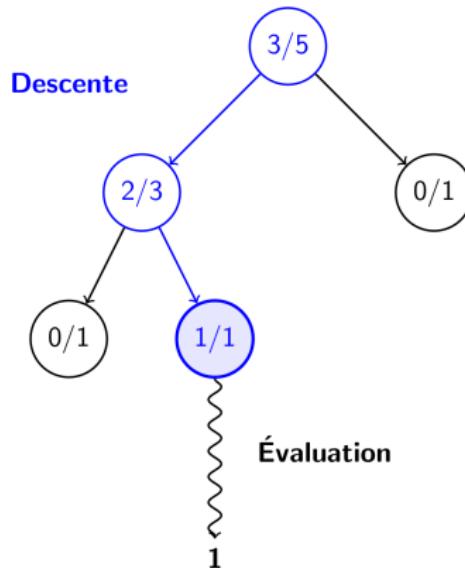
- **jeux :**

- jeux de plateaux
  - *Havannah* [Ewals, 2012], *Hex* [Arneson et al., 2009], ...
- jeux vidéos
  - *Ms. Pac-Man* [Pepels et al., 2014], ...
- jeux non déterministes
  - *Poker* [Rubin and Watson, 2011], *Magic* [Ward and Cowling, 2009], ...

- **autre :**

- *couplage de graphe* [Pinheiro et al., 2017]
- *planification* [Nakhost and Müller, 2009]
- *calcul de transformée de Fourier rapide* [De Mesmay et al., 2009]

## Conclusion



- méthode d'exploration d'arbre déséquilibré
- a permis de battre les meilleurs humains au jeu de **Go**
- méthode générique pouvant être appliquée à d'autres problèmes

## Références |

- Arneson, B., Hayward, R., and Henderson, P. (2009). Mohex wins hex tournament. *ICGA journal*, 32(2) :114.
- Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3) :235–256.
- Chaslot, G., Bakkes, S., Szita, I., and Spronck, P. (2008). Monte-carlo tree search : A new framework for game ai. In *AIIDE*.

## Références II

- Chaslot, G., Fiter, C., Hoock, J.-B., Rimmel, A., and Teytaud, O. (2009).  
Adding expert knowledge and exploration in monte-carlo tree search.  
In *Advances in Computer Games*, pages 1–13. Springer.
- De Mesmay, F., Rimmel, A., Voronenko, Y., and Püschel, M. (2009).  
Bandit-based optimization on graphs with application to library performance tuning.  
In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 729–736. ACM.
- Ewals, T. (2012).  
*Playing and solving Havannah*.  
PhD thesis, University of Alberta.

## Références III

- Gelly, S. and Silver, D. (2007).  
Combining online and offline knowledge in uct.  
In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM.
- Kocsis, L. and Szepesvári, C. (2006).  
Bandit based monte-carlo planning.  
In *European conference on machine learning*, pages 282–293. Springer.
- Lai, T. L. and Robbins, H. (1985).  
Asymptotically efficient adaptive allocation rules.  
*Advances in applied mathematics*, 6(1) :4–22.

## Références IV

- Lee, C.-S., Wang, M.-H., Chaslot, G., Hoock, J.-B., Rimmel, A., Teytaud, O., Tsai, S.-R., Hsu, S.-C., and Hong, T.-P. (2009).  
The computational intelligence of mogo revealed in taiwan's computer go tournaments.  
*IEEE Transactions on Computational Intelligence and AI in games*, 1(1) :73–89.
- Nakhost, H. and Müller, M. (2009).  
Monte-carlo exploration for deterministic planning.  
In *IJCAI*, volume 9, pages 1766–1771.
- Pepels, T., Winands, M. H., and Lanctot, M. (2014).  
Real-time monte carlo tree search in ms pac-man.  
*IEEE Transactions on Computational Intelligence and AI in games*, 6(3) :245–257.

## Références V

- Pinheiro, M. A., Kybic, J., and Fua, P. (2017).  
Geometric graph matching using monte carlo tree search.  
*IEEE transactions on pattern analysis and machine intelligence*, 39(11) :2171–2185.
- Rubin, J. and Watson, I. (2011).  
Computer poker : A review.  
*Artificial intelligence*, 175(5-6) :958–987.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016).  
Mastering the game of go with deep neural networks and tree search.  
*nature*, 529(7587) :484.

## Références VI

-  Ward, C. D. and Cowling, P. I. (2009).  
Monte carlo search applied to card selection in magic : The gathering.  
In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, pages 9–16. IEEE.