Jacques Pitrat
l'Intelligence Artificielle et les Jeux

Journée en hommage à Jacques Pitrat

6 Mars 2019

Tristan Cazenave

LAMSADE CNRS
Université Paris-Dauphine
PSL

Tristan.Cazenave@dauphine.psl.eu

Ma thèse avec Jacques Pitrat

Système d’apprentissage par auto-observation

Application au jeu de Go

13 décembre 1996

- Travaux de Jacques Pitrat sur les échecs :

  Pitrat, J. Realization of a Program Learning to Find Combinations at Chess. In Computer Oriented Learning Processes, Simon, J., Ed.,Noordhoff, 1976.
- Utilisation des règles du jeu pour généraliser de façon sûre.
- Repris par Steven Minton pour les Echecs [1984].
- Explanation Based Learning / Explanation Based Generalization [1986].

Ma thèse avec Jacques Pitrat

Système d’apprentissage par auto-observation

Application au jeu de Go

13 décembre 1996

- Apprentissage automatique de règles tactiques pour diriger la recherche arborescente.
- Les règles engendrées étaient des théorèmes du jeu de Go qui éliminaient de façon certaine les coups inutiles.
- Le système nommé Introspect s’inspirait d’une modélisation de la conscience réflexive.
- Le système partait des règles du jeu et s’améliorait automatiquement en jouant contre-lui même et en analysant son propre comportement.
- 5ème place sur plus de 40 participants à la FOST cup, IJCAI 1997, Nagoya.
- Bruno Bouzy et Indigo.

INFORMATION PROCESSING 68 - NORTH-HOLLAND PUBLISHING COMPANY - AMSTERDAM (1969)

REALIZATION OF A GENERAL GAME-PLAYING PROGRAM

JACQUES PITRAT
Institut Blaise Pascal, C.N.R.S.,
23, Rue du Maroc, 75, Paris XIX, France

We study some aspects of a general game-playing program. Such a program receives as data the rules of a game: an algorithm enumerating the moves and an algorithm indicating how to win. The program associates to each move the conditions necessary for this move to occur. It must find how to avoid a dangerous move.

We describe the part of the program playing the combinatorial game in order to win: how it can find the moves which lead to victory and what are the only opponent's moves with which he does not lose. This program has been tried with various games: chess, tic-tac-too, etc.

1. INTRODUCTION

My aim was to realize a program playing several games. The rules of the particular game which it must play are given as data. If we want to have a performing program, it must be capable of studying these rules.

The program is not completely general. It has limitations of three kinds:

a. It can only play games on a bidimensional board.
b. The rules of a game are written in a language which cannot describe every game, but which, however, covers a very large ground.
c. The more severe restrictions arise from heuristics which can be used in various games, but with very weak performances for some games.

I cannot describe the whole program, which is very large. I shall describe the combinatorial play which happens when we try to win, whatever the opponent may do.

The program can also play a positional game: this comes about when the opponent can play many moves without serious threats. We shall not discuss this part of the program.

2. LANGUAGE USED TO DESCRIBE THE RULES OF A GAME

There are two parameters for each square of the board:

a. One giving the occupation of the square: empty - friend - enemy,
b. One giving the type of man if the square is not empty.

For example, if the game is chess, the piece may be: king, rook, pawn ...

For some games, all men are of the same kind: tic-tac-toe, Go-Moku.

There are variables. They can represent a square or a number.

There are statements such as those of FORTRAN, ALGOL: arithmetic, test, go to statements.

Some are very specific to games:

a. Result statement. This statement gives information about winning in a particular state of the board. This may be: victory, loss, draw, no victory ...
b. Move statement. This statement describes a move, which can be made up of several parts (partial moves). The parts of a move fall into four types:
   i. The man in square A goes to square B
   ii. The man in square A is captured
   iii. A man of type T is put in square A
   iv. The man in square A becomes a new type T.

To sum up, the rules of a game are given as algorithms written in the language described above: an algorithm enumerating legal moves and an algorithm indicating how to win.

3. STUDY OF AN ALGORITHM

First, the program must find, for each move or for each indication of victory, the conditions necessary to obtain it. This is useful if we want to destroy an opponent's move or if we want to try to make a move possible, or to win, or to escape a danger.

These conditions are not given by algorithms,


```
6,1 - 3,4     If  7,8 - 6,8
              Then  3,4 - 6,7 and win
              Thus  7,8 - 8,8

7,5 - 6,7           8,8 - 7,8
6,7 - 8,6     If  7,8 - 6,8
              Then  3,4 - 6,7 and win
              Thus  7,8 - 8-8

3,4 - 7,8       x  7,8 ; 1,8 - 7,8

8,8 - 6,7     and win

              Time : 77 seconds
```

Fig. 4.

only one game. Artificial intelligence aims to get general programs. If they are too particular, we often have progress in the theory of the particular problem, but no progress in the theory of artificial intelligence. Also, a particular program cannot study new problems.

This program has been tried on games as varied as tic-tac-toe, Go-Moku, chess and cylindrical chess.

It is important to see that nowhere does it use the particular properties of the game. If it is playing chess, it does not know, for example, that it is advisable to protect its men.

REFERENCE

[1] Baylor and Simon, A chess mating combinations program, AFIPS, Vol. 28 p. 341.

DISCUSSION

Question by K. Paton

In your conclusion, you state that in playing chess, the program does not know that it is protecting men. Does the program ever find out that it is protecting men?

Answer

No, it cannot learn. The general method enables it to perform in each specific case, but it is not formally capable of discovering the notion of protection. Incidentally, the idea of protection in chess does not apply in all cases, and makes the formal discovery of the program difficult.

Question by D. Levy

How does the program decide what to do in a non-tactical position?

Answer

The program takes into account the possibility of winning and forcing, e.g. in Go-Molar, it takes into account relationships of moves and opponent's moves, and tries to increase its mobility (i.e. the number of possible moves) and lessen the mobility of the opponent. The definition of mobility is included in the program, and is completely general.

Question by P. Braffort

I can understand the advantage of a general approach to the problem of artificial intelligence treating games of a different nature on the same level and with the same formalism for the establishment of rules. But have you also made an independent evaluation or comparison of your program with others, and if so how do you find it?

Answer

The aim of the program is to play several board games, but not specifically any one game, and I would not expect it to be better than a specific program for a specific game.

Question by G. Nagy

Can you compare the performance of your program with the Simon-Baylor results?

Answer

A general purpose game playing program will obviously be less efficient than a special program for a game. The combinational program is finished, and I am now writing the positional program.

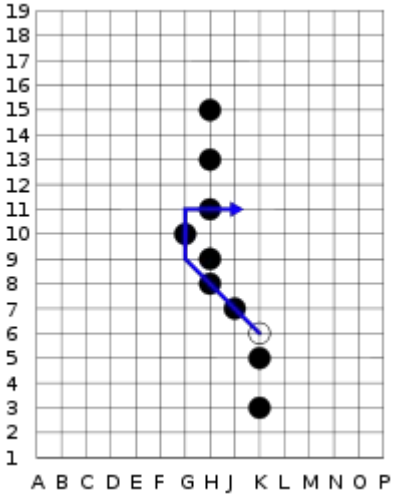
General Game Playing

- 1968 : Jacques Pitrat invente le General Game Playing.
- 1997 : Deep Blue bat Kasparov aux échecs mais il ne sait pas jouer à d'autres jeux.
- 2005 : Michael Genesereth propose le GDL et des compétitions de GGP.
- Compétition annuelle à AAAI organisée par Stanford.
- Utilisation d'une représentation logique du premier ordre pour représenter les jeux.
- Jeux à deux joueurs, multi-joueurs et à un joueur.
- On donne aux programmes la description GDL d'un jeu juste avant d'y jouer.
- Ary [Méhat et al.] champion du monde IJCAI 2009 et AAAI 2010.

Berder

Le Football des Philosophes

- Jean Méhat et John Conway.
- Tournoi de programmes de Phutball à Berder en 1999.
- Politique (Jean-Yves Lucas et Hélène Giroire) ou évaluation (Jacques Pitrat et al.) ?
- Mobilité versus sélectivité sur les coups.
- Résolution du Phutball 9x9.



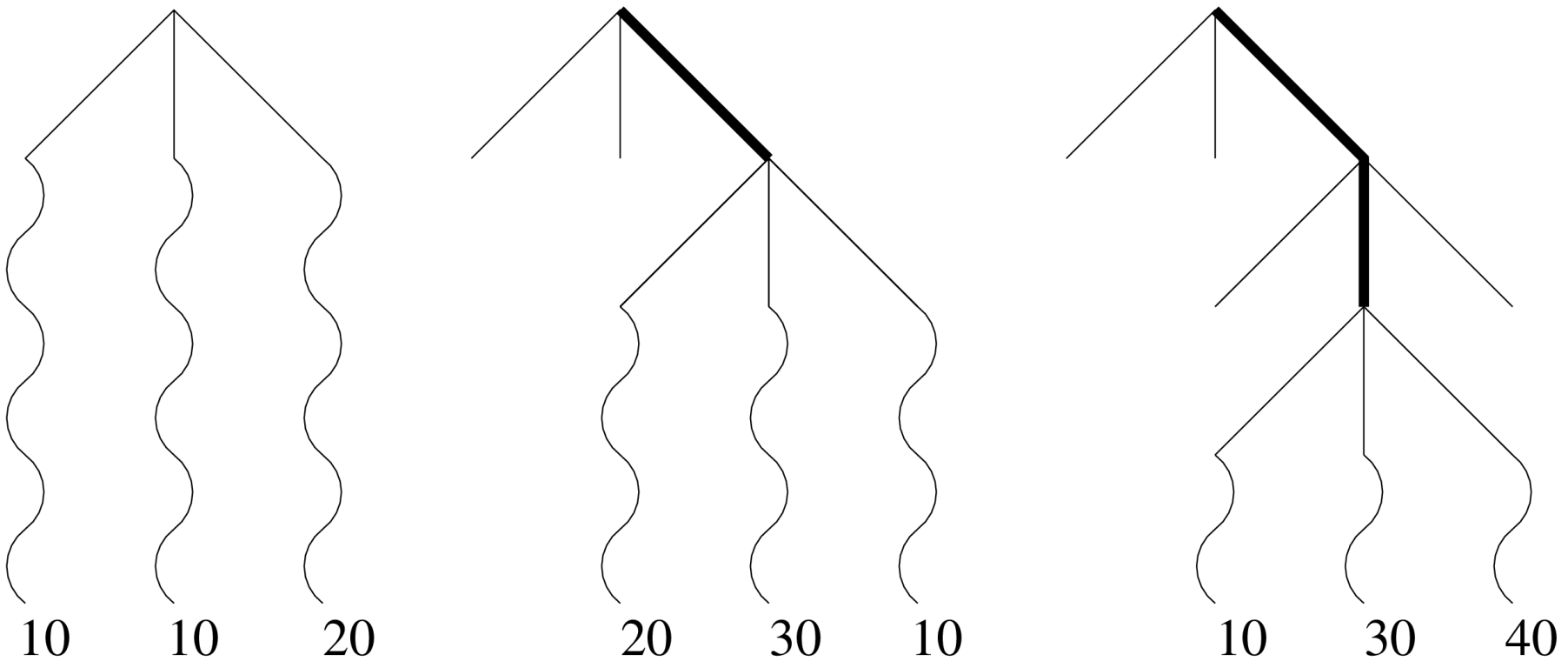
Bootstrap

- Principe : utiliser le système pour améliorer le système.
- Bootstrapper l'IA = faire une IA qui fait de l'IA.
- Maciste et les métaconnaissances [Pitrat 1990]
- Malice le solveur [Pitrat 1995-2009]
- CAIA le Chercheur Artificiel en Intelligence Artificielle.

Bootstrap

- La recherche Monte-Carlo a révolutionné la programmation du Go puis des jeux.
- Jacques Pitrat l’appelait le Random Go :)
- Application des idées de Jacques Pitrat à la recherche Monte-Carlo.
- Utiliser la recherche Monte-Carlo pour améliorer la recherche Monte-Carlo.
- Nested Monte Carlo Search [Cazenave 2009].

Nested Monte-Carlo Search



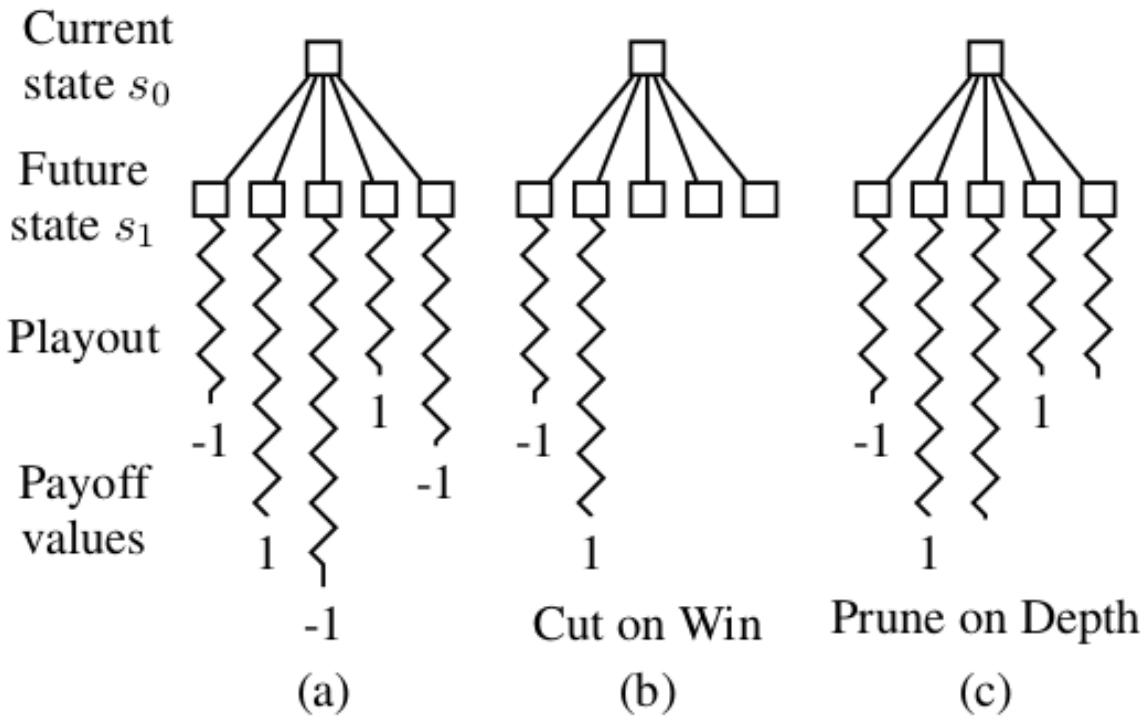
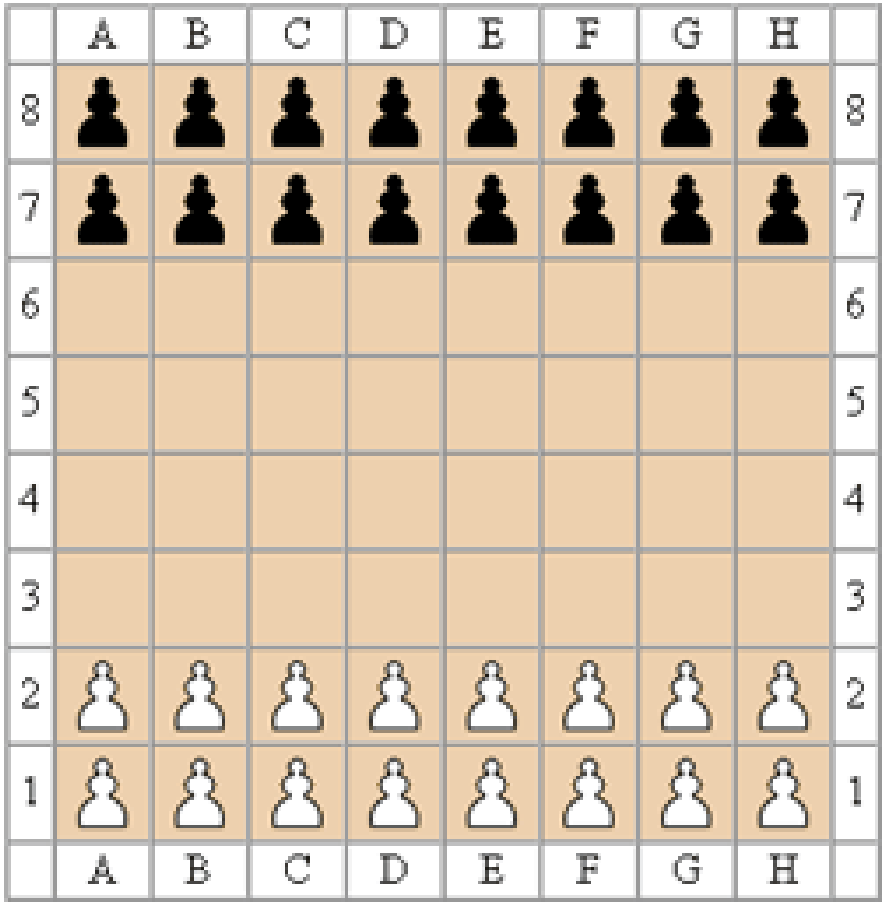


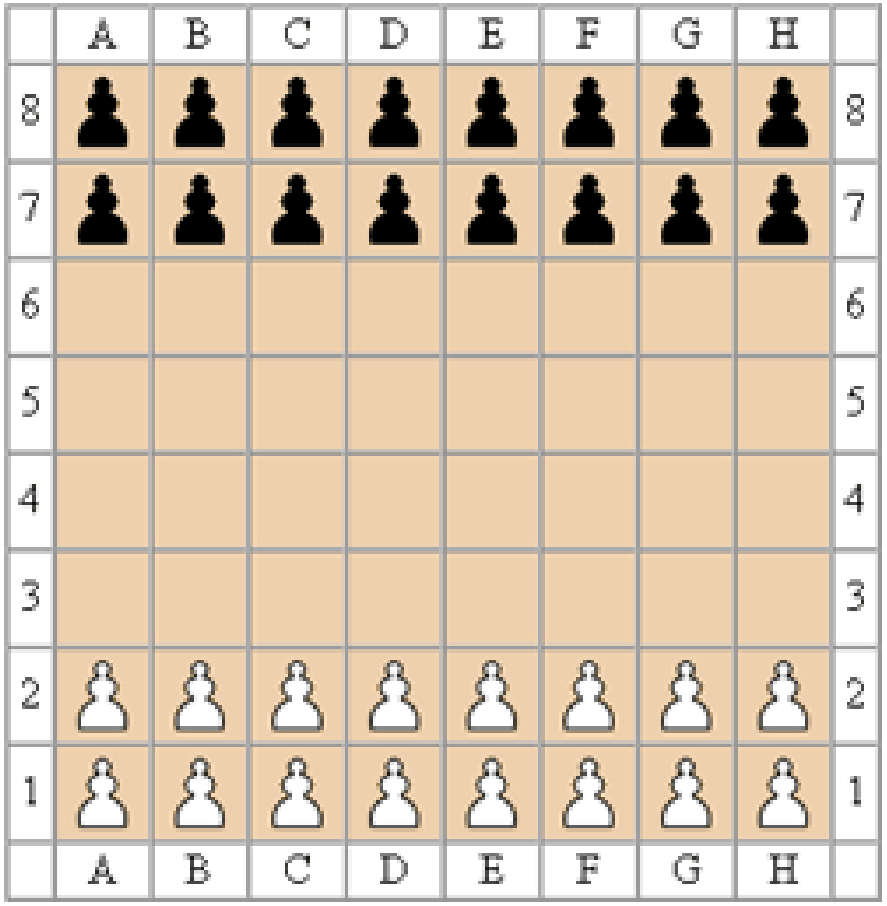
Figure 2: Effect of the pruning strategies on an NMCS run. We assume a max root state and a left-to-right evaluation order. (a) In the standard case, the second and the fourth successor states are equally preferred. With discounting, the fourth successor state is the most-preferred one. (b) This fourth state may fail to be selected when Cut on Win is enabled. (c) With Pruning on Depth and discounting, however, this fourth state would be found and preferred too.

Breakthrough



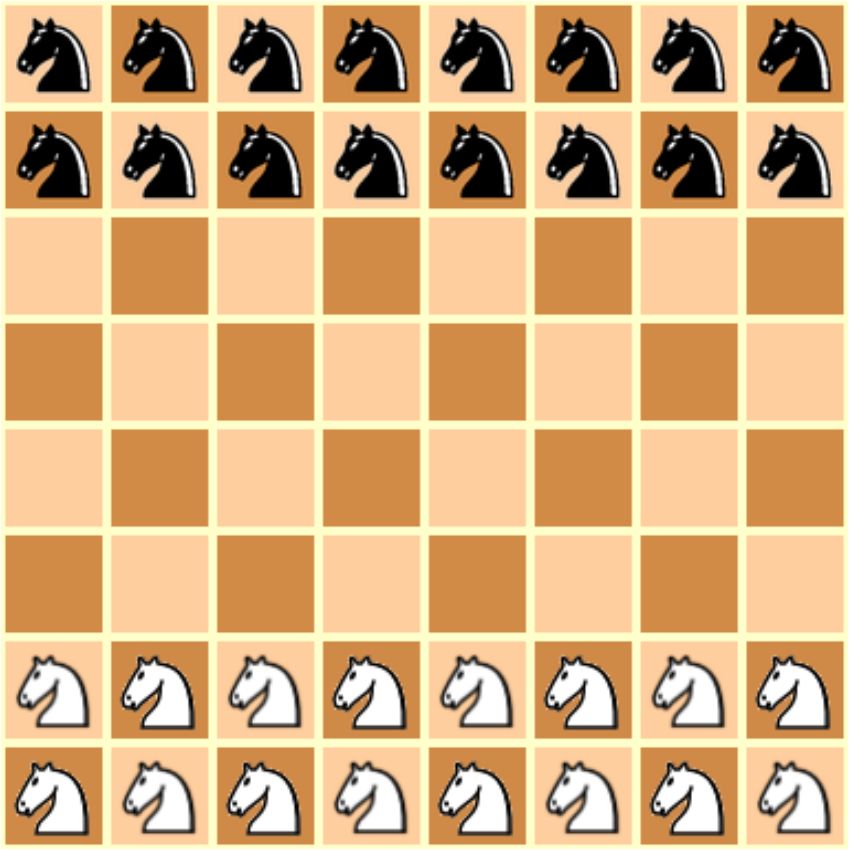
- The first player to reach the opposite line has won

Misère Breakthrough



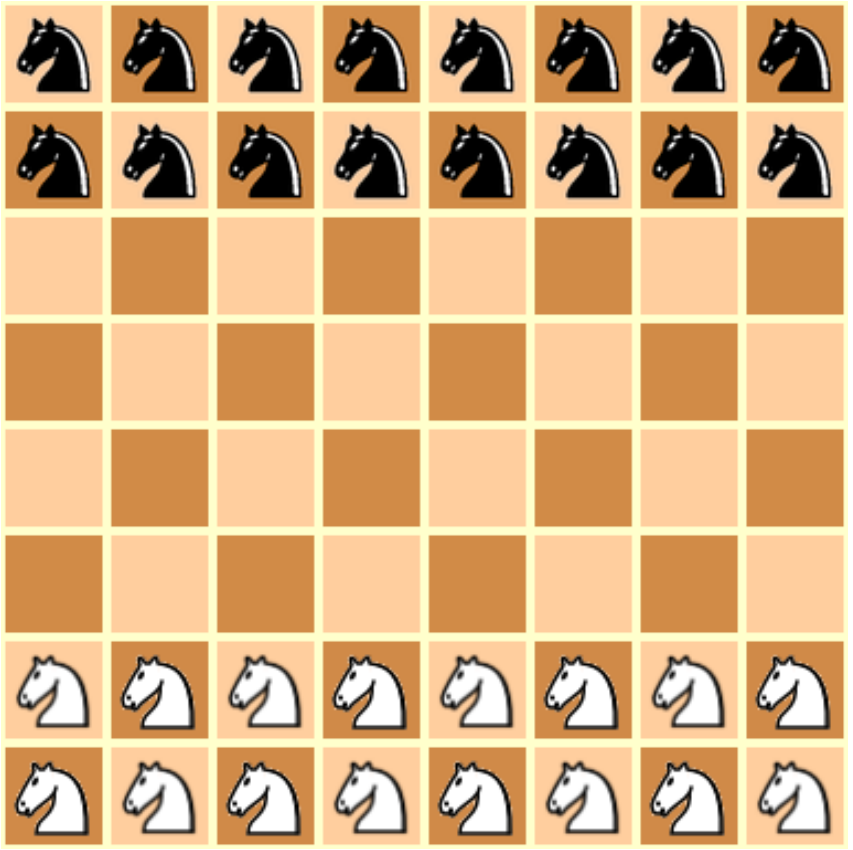
- The first player to reach the opposite line has lost

Knightthrough



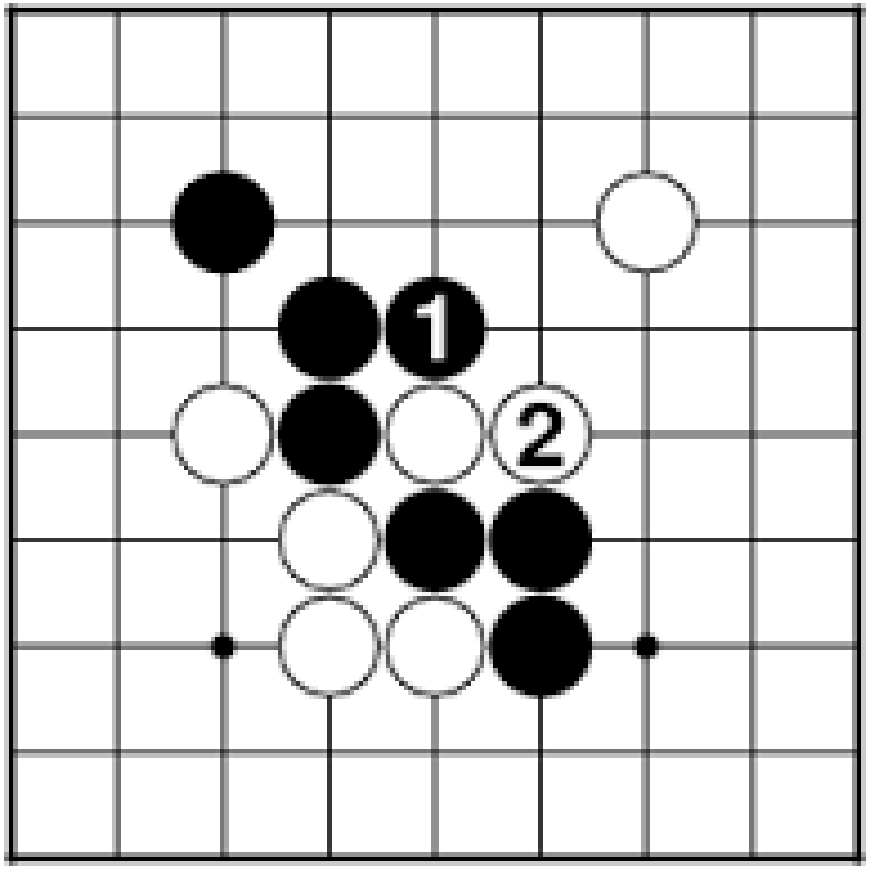
- The first to put a knight on the opposite side has won.

Misère Knightthrough



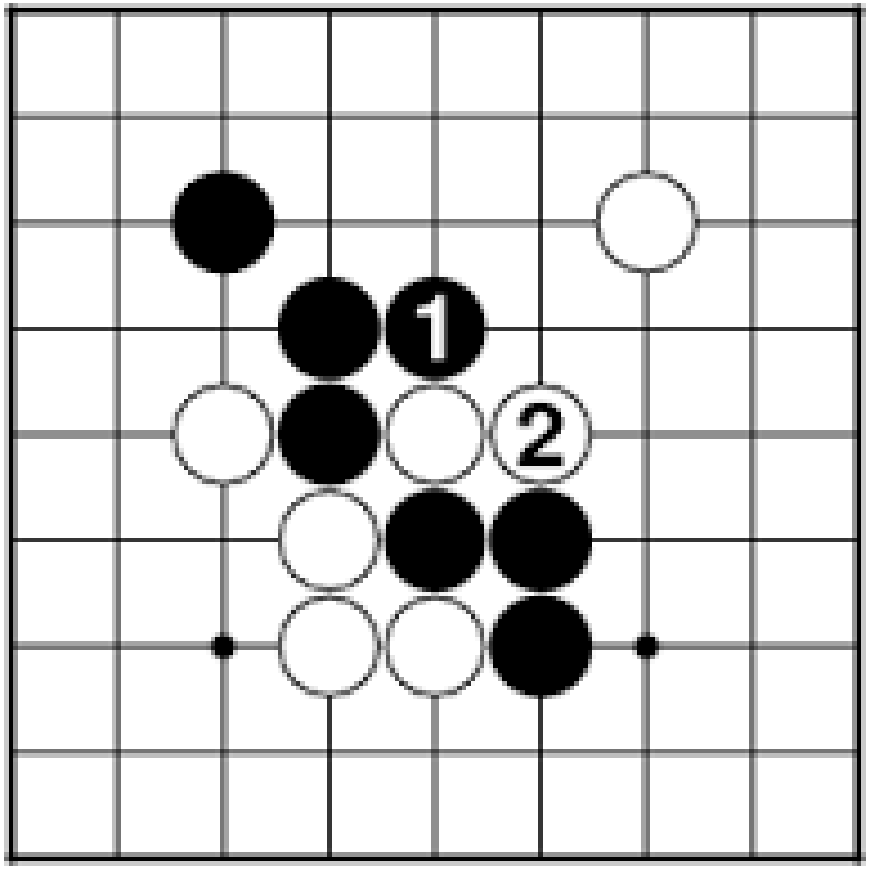
- The first to put a knight on the opposite side has lost.

Atarigo



- The first to capture has won

Nogo



- The first to capture has lost

Domineering
Misère Domineering

- The last to play has won / lost.

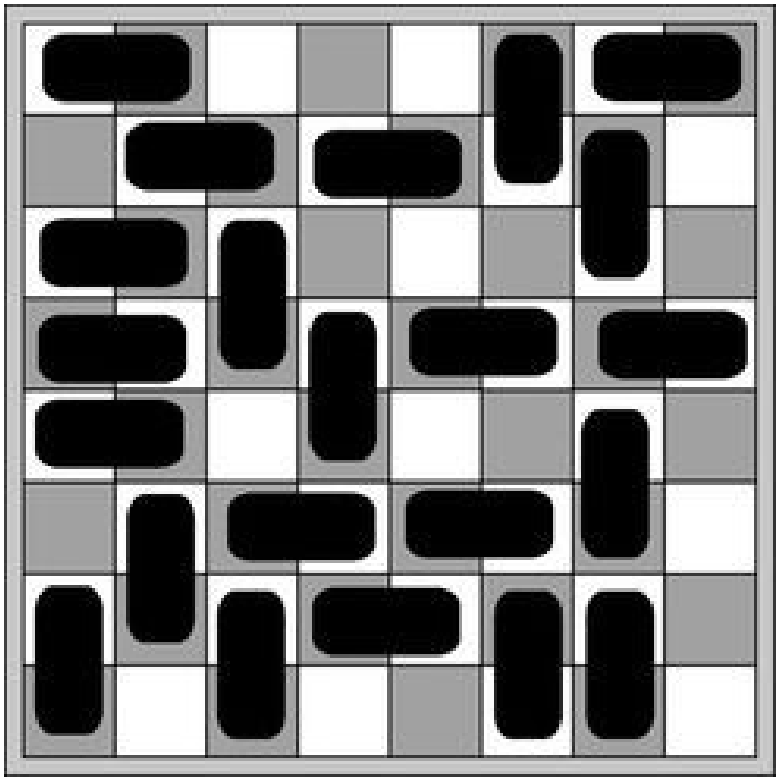


Table 4: Win percentages of NMCS against a standard MCTS player for various settings and thinking times.

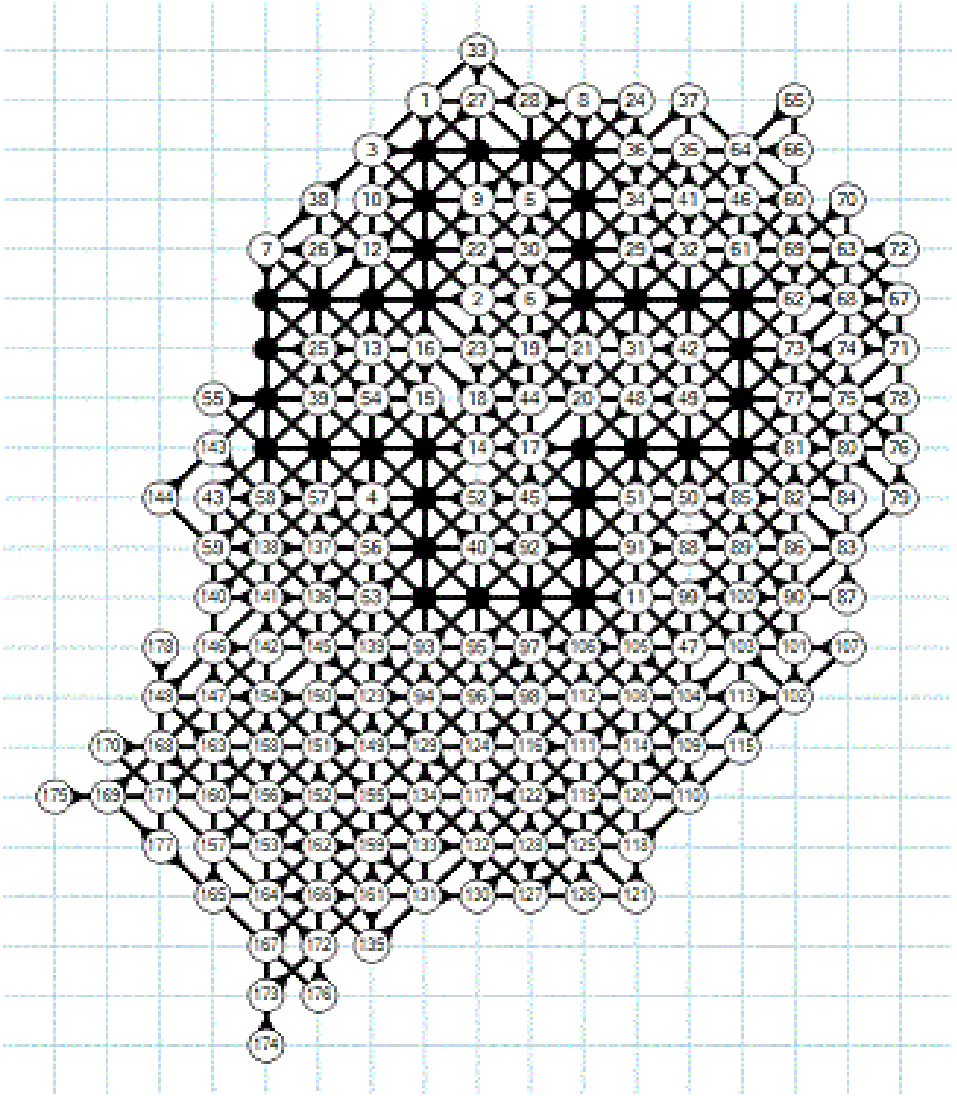
Game		n	COW	POD	10ms	20ms	40ms	80ms	160ms	320ms
Breakthrough		1			3.2	6.0	12.0	11.6	7.8	6.4
		1		✔	27.6	22.6	16.8	21.6	15.4	20.4
		1	✔		22.6	25.2	30.4	34.6	35.2	39.6
		2	✔		4.6	2.0	2.4	1.4	2.4	3.8
	misère	1			85.4	83.4	70.2	60.8	57.0	56.4
	misère	1		✔	91.4	95.6	97.0	97.8	98.8	98.8
	misère	1	✔		95.2	95.2	98.0	99.0	99.8	**99.8**
	misère	2	✔		1.0	27.6	43.6	87.0	93.2	95.6
Knightthrough		1			42.2	57.2	9.8	49.4	50.2	50.0
		1		✔	68.6	50.2	42.4	42.4	46.4	44.6
		1	✔		27.2	25.4	28.0	43.4	49.2	49.6
		2	✔		20.0	16.4	5.8	1.8	29.2	38.2
	misère	1			43.0	31.6	20.0	15.4	11.2	12.6
	misère	1		✔	54.6	72.2	80.6	88.4	94.2	98.4
	misère	1	✔		77.8	82.2	88.8	94.4	98.2	**98.6**
	misère	2	✔		20.8	18.6	32.2	42.2	54.0	67.0
Domineering		1			13.4	8.6	8.6	6.0	14.2	28.0
		1		✔	40.8	34.4	37.4	48.4	50.0	50.0
		1	✔		44.4	38.6	40.6	49.4	50.0	50.0
		2	✔		11.2	14.4	20.2	25.2	32.2	45.4
	misère	1			33.4	25.2	20.0	18.8	13.2	12.2
	misère	1		✔	45.4	47.2	56.8	60.2	62.8	54.2
	misère	1	✔		69.4	66.6	71.6	70.4	68.4	**58.6**
	misère	2	✔		37.0	45.2	45.6	51.0	57.8	53.6
NoGo		1			5.8	3.0	2.6	3.0	0.6	0.8
		1		✔	7.2	16.0	31.8	35.2	35.4	40.6
		1	✔		37.6	39.2	38.4	40.8	47.8	48.0
		2	✔		0.4	2.8	5.4	15.0	20.6	17.0
	misère	1			14.6	6.6	5.2	3.0	2.4	1.8
	misère	1		✔	17.2	25.0	38.8	51.2	48.2	48.8
	misère	1	✔		55.4	56.6	57.0	57.6	54.6	**60.8**
	misère	2	✔		5.2	10.6	19.4	35.6	37.2	47.8
Atari-Go		1			0.6	2.2	4.6	5.4	6.8	7.6
		1		✔	0.2	19.2	42.0	42.0	55.4	67.2
		1	✔		42.0	59.0	60.2	71.0	71.2	**77.2**
		2	✔		0.2	0.0	0.6	7.4	8.6	4.8

Applications

Nested Monte Carlo Search :

- Morpion Solitaire [Cazenave 2009]
- SameGame [Cazenave 2009]
- Sudoku [Cazenave 2009]
- Expression Discovery [Cazenave 2010]
- The Snake in the Box [Kinny 2012].
- Cooperative Pathfinding [Bouzy 2013].
- Software Testing [Poulding et al. 2014]
- Heuristic Model-Checking [Poulding et al. 2015]
- Pancake problem [Bouzy 2015]
- Games [Cazenave et al. 2016]
- Cryptography [Dwivedi et al. 2018]
- RNA inverse folding problem [Portela 2019]
- …

Morpion Solitaire



World record [Rosin 2011]

Applications

Nested Rollout Policy Adaptation :

- Morpion Solitaire [Rosin 2011]
- CrossWords [Rosin 2011]
- Travelling Salesman Problem with Time Windows [Cazenave et al. 2012]
- 3D Packing with Object Orientation [Edelkamp et al. 2014]
- Multiple Sequence Alignment [Edelkamp et al. 2015]
- SameGame [Cazenave et al. 2016]
- Vehicle Routing Problems [Edelkamp et al. 2016]
- Graph Coloring [Cazenave et al. 2020]
- ...

Bootstrap

- AlphaGo [Silver et al. 2016] : apprentissage supervisé pour la politique puis self play pour la valeur.
- AlphaGo Zero [Silver et al. 2017] : la politique et la valeur sont bootstrappées en faisant jouer la politique et la valeur contre elles mêmes.
- Alpha Zero [Silver et al. 2018] : Go, Echecs, Shogi.
- Polygames [Cazenave et al. 2020] : Bootstrap de l'IA pour tous les jeux.
- Bootstrap et General Game Playing surhumain !

Conclusion

- Les idées originales de Jacques Pitrat sur L'IA et les jeux, Bootstrap et General Game Playing, sont au cœur des systèmes actuels de jeux qui ont des performances bien meilleures que les êtres humains.
- Les applications dépassent largement le cadre des jeux.