# MCTS – Monte Carlo Tree Search in Chess

Monte Carlo (dzielnica Monako)
jest historyczną stolicą **hazardu**.

~1950 Shannon

Turnieje programów do Go:

- **2006** Wygrywa **Crazy Stone**
- 2007 Wygrywa MoGo

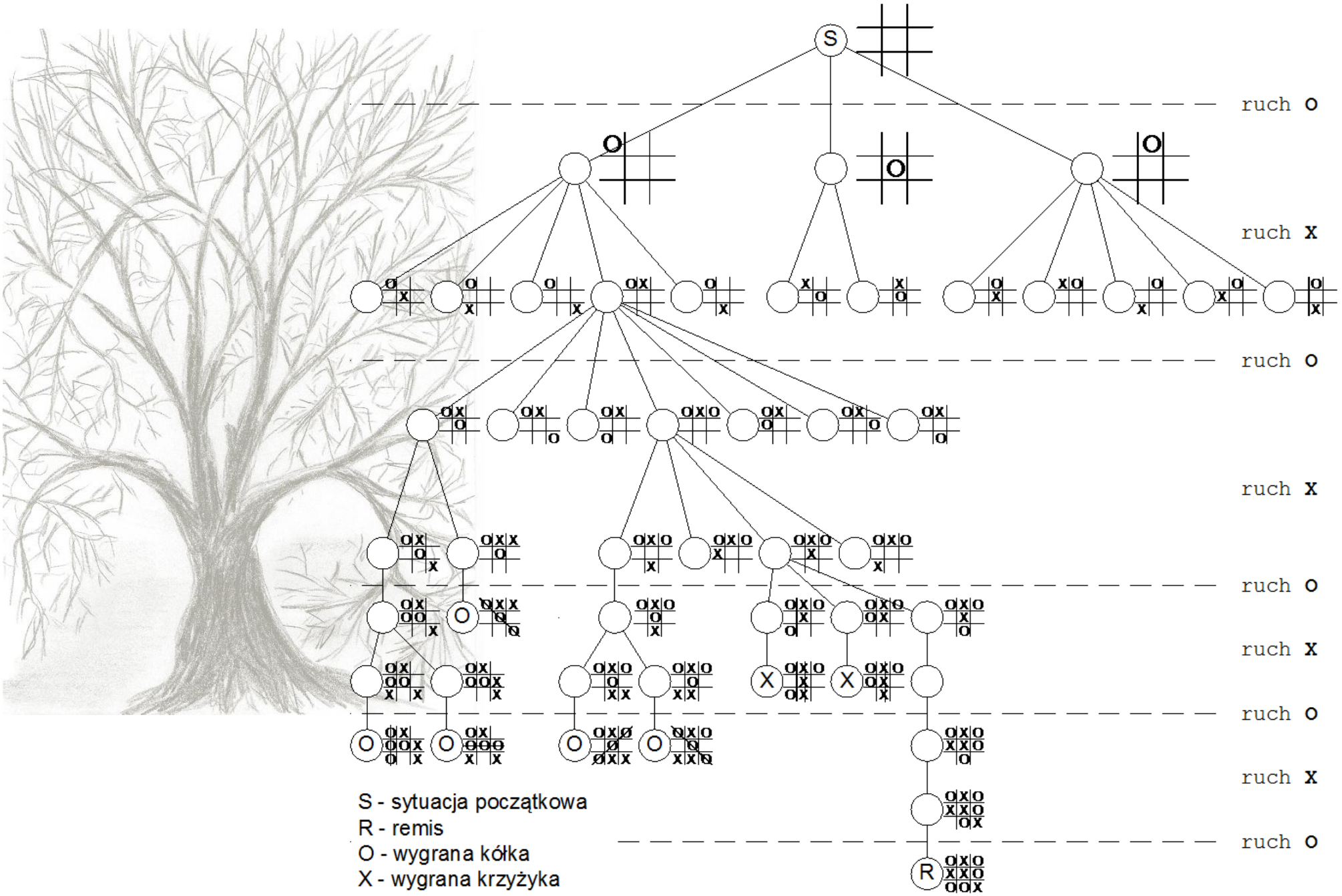
(oba używają MCTS)

Szkic prezentacji, wersja 0.1

# Dlaczego programowanie szachów?

- Intelektualne wyzwanie
  – wciąż człowiek pokonuje komp., techniki szachowe przenosi się na trudniejsze gry
- Niezbędna abstrakcja i realny wpływ na świat
  – szachy są łatwe do zdefiniowania
  – algorytmy i technologie rozwijane na potrzeby szachów często służą do zmieniania świata na lepsze (superkomputery, alg. przeszukiwania używane są w przemyśle)
- Precyzyjne zasady i cel gry
  – wiadomo od czego zacząć, zwykle wiadomo w jakim kierunku podążać
- Szybkie sprzężenie zwrotne i jasne rezultaty
  – łatwo mierzy się skuteczność wprowadzonych zmian, obiektywny ranking ELO
- Zabawa, przyjemność satysfakcja
  – od dziecka ludzie uwielbiają różne gry

# Drzewo Gry


S
ruch O
ruch X
ruch O
ruch X
ruch O
ruch X
ruch O
ruch X
ruch O
S - sytuacja początkowa
R - remis
O - wygrana kółka
X - wygrana krzyżyka

# Myślenie człowieka vs komputer
# Mur pionów

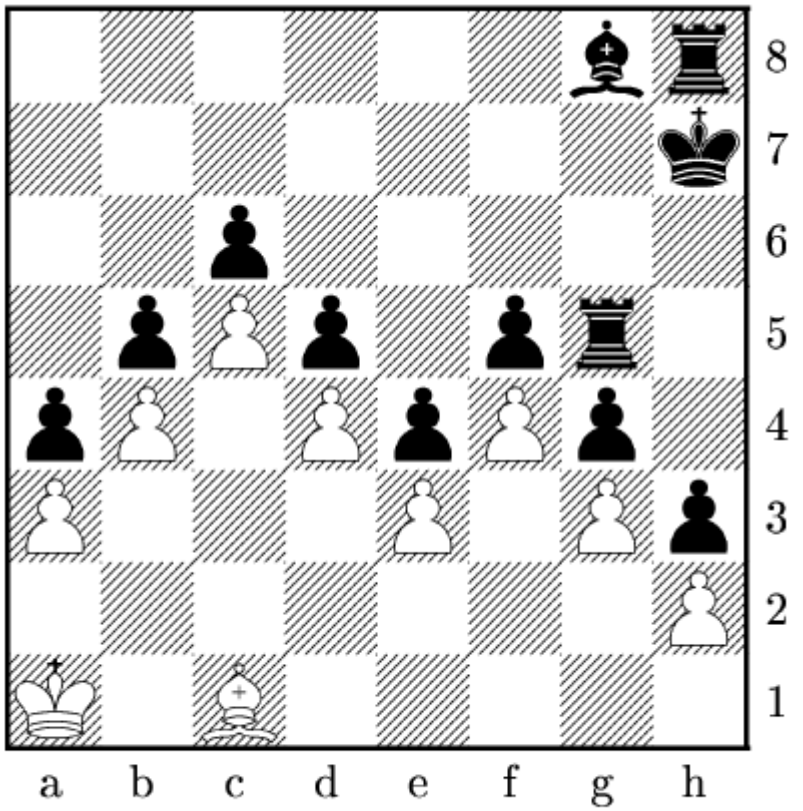


Diagram 5. Mur pionów.

# Brutalna siła

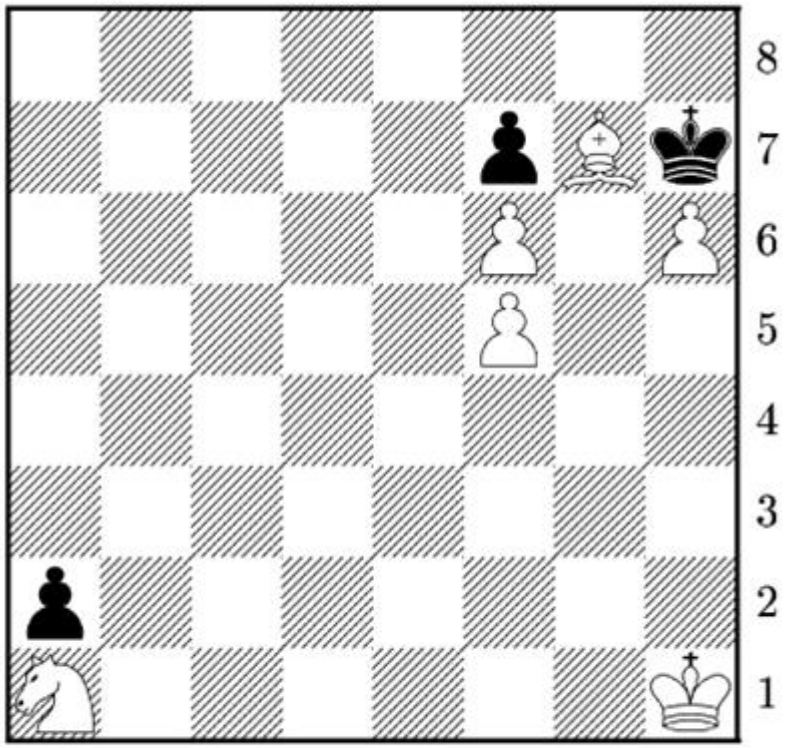


Diagram 6. Uwolnienie skoczka.

## Człowiek widzi:

- Biały skoczek jest zablokowany
- Białym królem należy zbić piona co uwolni skoczka
- W 12 ruchu skoczek asystuje przy macie zadanym pionem h6-h7++

Komputer musi przeszukać

**23** półruchy w głąb!

# MCTS nadaje się do …

Do trudnych gier:

Go, Hex, Szachy, …

Do gier z elementami losowymi:

Poker, Kości,
Osadnicy z Catanu, …

Do symulacji ekonomicznych

Do symulacji biologicznych

Do planowania

…

## Monte-Carlo Tree Search in Settlers of Catan

István Szita[1], Guillaume Chaslot[1], and Pieter Spronck[2]

[1] Maastricht University, Department of Knowledge Engineering
[2] Tilburg University, Tilburg centre for Creative Computing
szityu@gmail.com, g.chaslot@micc.unimaas.nl, p.spronck@uvt.nl

**Abstract.** Games are considered important benchmark tasks of artificial intelligence research. Modern strategic board games can typically be played by three or more people, which makes them suitable test beds for investigating multi-player strategic decision making. Monte-Carlo Tree Search (MCTS) is a recently published family of algorithms that achieved successful results with classical, two-player, perfect-information games such as Go. In this paper we apply MCTS to the multi-player, non-deterministic board game Settlers of Catan. We implemented an agent that is able to play against computer-controlled and human players. We show that MCTS can be adapted successfully to multi-agent environments, and present two approaches of providing the agent with a limited amount of domain knowledge. Our results show that the agent has considerable playing strength when compared to existing heuristics for the game. We conclude that MCTS is suitable for implementing a strong Settlers of Catan player.



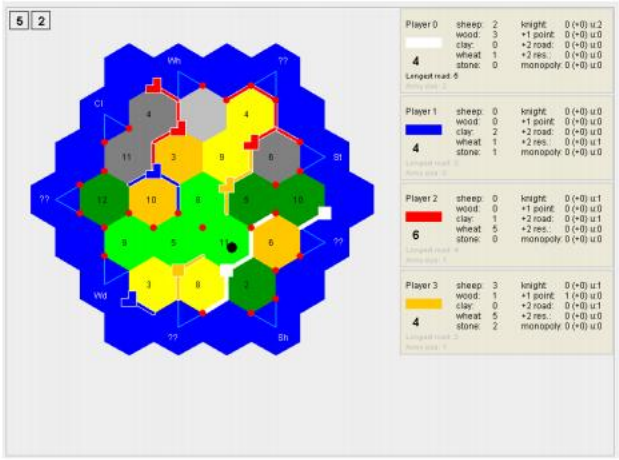


**Fig. 1.** A game state of Settlers of Catan, represented in our program SmartSettlers

# Zalety MCTS

- Wystarczą zasady gry
- Nie potrzeba funkcji oceny pozycji, wiedzy taktycznej czy strategicznej
- Asymetria – drzewo rośnie tam gdzie trzeba
- Równoległość
- Przejrzysty algorytm naśladujący myślenie człowieka
- Kontrola czasu

# Wady MCTS

- Całe drzewo Monte Carlo trzymamy w pamięci (na szczęście nie jest duże)
- Słabo działa w sytuacjach gdy trzeba wszystko postawić na jedna kartę (np. poświęcenie)
- Wymaga dużej ilości iteracji

# Cechy MCTS

- Lepsze w grach **strategicznych** niż w taktycznych

# Podstawy MCTS

- Selekcja
- Ekspansja – rozwijanie drzewa
- Losowa symulacja
- Propagacja wsteczna wyników

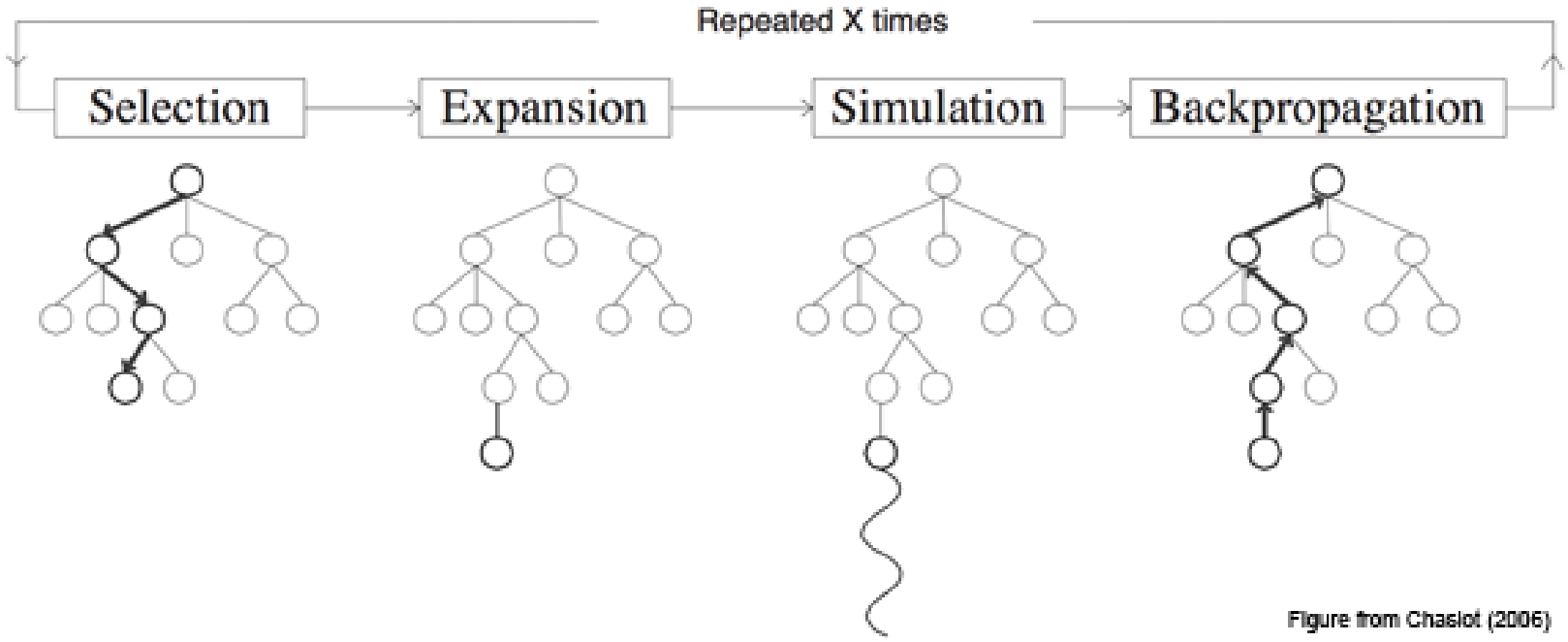
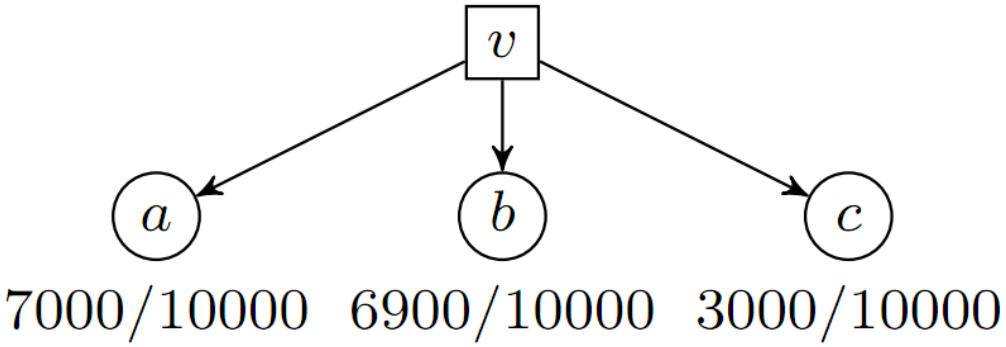



Figure from Chaslot (2006)

# Granica ufności

- Dla każdego stanu przeprowadziliśmy 10 000 losowych rozgrywek.
- Wyraźnie ruch c jest gorszy niż ruchy a i b.
- Po co tak dokładnie go badać?




Wartość oczekiwana oceny sytuacji wynosi: $E_v = \frac{w_v}{n_v}.$

Interesuje nas przedział taki, że prawdopodobieństwo, że rzeczywista ocena należy do tego przedziału, jest dosyć duże:

$$\Pr(x \in [E_v - c\sigma_v, E_v + c\sigma_v]) \geq 1 - \varepsilon \text{ dla } \sigma_v^2 = \frac{1}{n_v},$$

Przedział $[E_v - c\sigma_v, E_v + c\sigma_v]$ nazywamy przedziałem ufności.

# Algorytm MCTS (1)

**Algorytm 4** Wybór syna metodą granicy ufności.

1: **function** GranicaUfności($v, s$)
2: $E_s \leftarrow \frac{w_s}{n_s}$, $\sigma_s \leftarrow \sqrt{\frac{\log n_v}{n_s}}$
3: **if** $v$ jest max **then**
4: **return** $E_s + \sigma_s$
5: **else** ▷ $v$ jest min
6: **return** $E_s - \sigma_s$
7: **function** WybierzSyna($v$)
8: **return** syn $s$ stanu $v$ optymalizujący wartość GranicaUfności($v, s$)

# Algorytm MCTS (2)

**Algorytm 5** Przeszukiwanie drzewa gry Monte Carlo.

```
1:  function ROZGRYWKA(v)
2:      if v jest liściem then
3:          if n_v ≥ N_0 then
4:              rozwiń v
5:              W ← ROZGRYWKA(WYBIERZSYNA(v))
6:          else
7:              Przeprowadź rozgrywkę do końca, losowo wybierając ruchy
8:              return 1 dla wygranej, bądź 0 dla przegranej
9:      else
10:         W ← ROZGRYWKA(WYBIERZSYNA(v))
11:     n_v ← n_v + 1, w_v ← w_v + W
12:     return W
13: function ZNAJDŹNAJLEPSZYRUCH(v)
14:     utwórz korzeń drzewa ze stanem v
15:     rozwiń v                          ▷ korzeń wyjątkowo rozwijamy od razu
16:     while nie skończył się nam czas na ruch do
17:         ROZGRYWKA(v)
18:     if v jest max then
19:         return syn s stanu v maksymalizujący E_s = w_s / n_s
20:     else ▷ v jest min
21:         return syn s stanu v minimalizujący E_s = w_s / n_s
```

# Materiały (updated 3/2011)

**Polskie www:**

Wikipedia
- http://pl.wikipedia.org/wiki/Szachy_komputerowe
- http://pl.wikipedia.org/wiki/Algorytm_min-max
- http://pl.wikipedia.org/wiki/Algorytm_alfa-beta

Szachy i komputery (artykuł, 23 rozdziały)
- http://pclab.pl/art34801.html

Polskie programy szachowe (21 linków)
- http://lpps.maciej.szmit.info/programy.html

**Prace magisterskie:**

Program Joanna, 68s, źródła w C - Adam Kujawski
- http://www.apsys.waw.pl/joanna2002/

Samouczenie programów szachowych, 54s - Tomasz Michniewski
- http://www.armageddon.szach.pl/pracamag.pdf

Design and Implementation of the Rookie 2.0 Chess Playing Program, 89s – van Kervinck
- http://alexandria.tue.nl/extra2/afstversl/wsk-i/kervinck2002.pdf

**Inne materiały:**
- http://scholar.google.pl/scholar?q=computer+chess
- http://scholar.google.pl/scholar?q=chess+programming

**Zagraniczne www:**

Wikipedia
- http://en.wikipedia.org/wiki/Computer_chess
- http://en.wikipedia.org/wiki/Minimax
- http://en.wikipedia.org/wiki/Alpha-beta_pruning
- http://en.wikipedia.org/wiki/Game_tree

Chess Programming Wiki
- http://chessprogramming.wikispaces.com/

Chess Programming Introduction:
- http://www.gamedev.net/reference/programming/features/chess1/
- http://www.frayn.net/beowulf/theory.html
- http://www.maths.nott.ac.uk/personal/anw/G13GT1/compch.html

Strony z linkami:
- http://www.chessopolis.com/cchess.htm (~100)
- http://u.cs.biu.ac.il/~davoudo/tutorials.html (~50)
- http://wbec-ridderkerk.nl/html/ProgrLinks.html (~40)
- http://www.chessbin.com/ (~40, C# starter)
- http://www.lkessler.com/cclinks.shtml (~150)

Monte Calro
- http://www.mcts.ai/about/index.html
- http://www.mimuw.edu.pl/delta/artykuly/delta2010-07/2010-07-1.pdf
- http://www.mimuw.edu.pl/~pan/gry.pdf

Programy
- http://chessprogramming.wikispaces.com/Engines Lista
- http://www.rybkachess.com/ Rybka
- http://www.tckerrigan.com/Chess/TSCP TSCP

Algorytmy:
- http://chessprogramming.org/ Parallel Alpha-Beta Search
- http://u.cs.biu.ac.il/~davoudo/pubs/vrfd_null.html Null-Move Pruning

# Demo