Przeszukiwanie w grach

Paweł Rychlikowski

Instytut Informatyki UWr

9 kwietnia 2024

Przykładowa gra

- Gracz **A** wybiera jeden z trzech zbiorów:
  1. $\{-50, 50\}$
  2. $\{1, 3\}$
  3. $\{-5, 15\}$
- Następnie gracz **B** wybiera liczbę z tego zbioru.

Pytanie

Co powinien zrobić **A**, żeby uzyskać jak największą liczbę?

Przykładowa gra

Nasza gra

1. $\{-50, 50\}$
2. $\{1, 3\}$
3. $\{-5, 15\}$

Racjonalny wybór dla **A** zależy od (modelu) gracza **B**

- Współpracujący: Oczywiście 1.
- Losowy (z $p = \frac{1}{2}$)) Wybór 3 (średnio 5)
- „Złośliwy”:wybór 2 (gwarantujemy wartość 1)

Wyszukiwanie w grach

- Nieco inna rodzina zadań wyszukiwania, w których mamy dwóch (lub więcej) agentów.
- Interesy agentów są (przynajmniej częściowo) rozbieżne.
- Rozgrywka przebiega w turach, w których gracze na zmienę wybierają swoje ruchy.

Definicja gry

Definicja

Gra jest problemem przeszukiwania, zadanym przez następujące składowe:

1. Zbiór stanów, a w nim S_0, czyli stan początkowy
2. player(s), funkcja określająca gracza, który gra w danym stanie.
3. actions(s) – zbiór ruchów możliwych w stanie s
4. result(s,a) – funkcja zwracająca stan powstały w wyniku zastosowania akcji a w stanie s.
5. terminal(s) – funkcja sprawdzająca, czy dany stan kończy grę.
6. utility(s, player) – funkcja o wartościach rzeczywistych, opisująca wynik gry z punktu widzenia danego gracza.

Gra o sumie zerowej

Definicja

W grze o sumie zerowej suma wartości stanów terminalnych dla wszystkich graczy jest stała (niekonieczne zera, ale...)

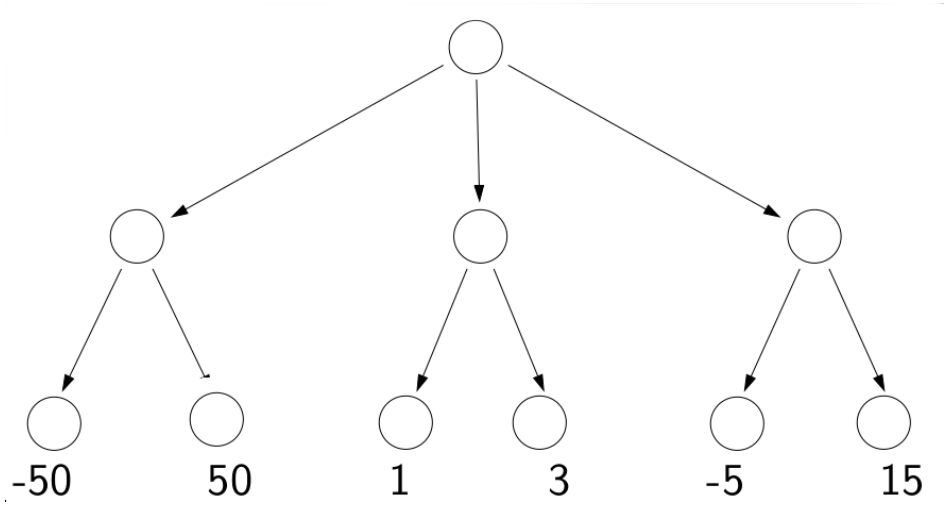
Konsekwencje:

- Zysk jednego gracza, jest stratą drugiego.
- Kooperacja nic nie daje.

Uwaga

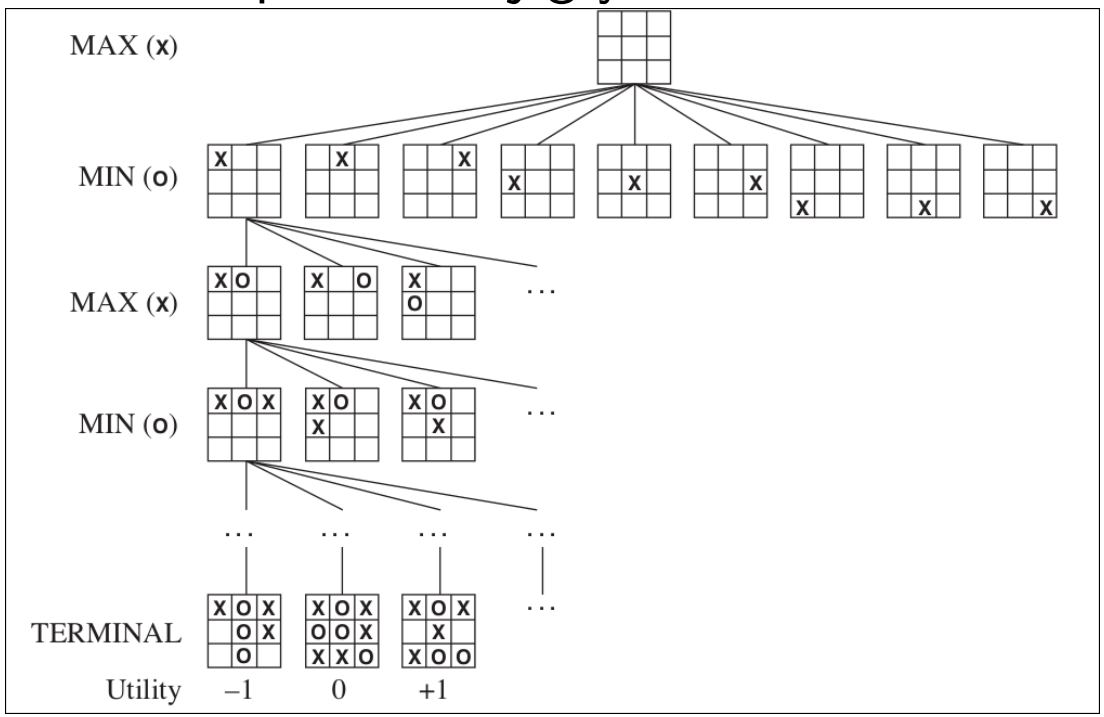
Zaczniemy od gier o sumie zerowej i gracza, wcześniej nazwanego złośliwym (lepiej go nazwać racjonalnym)

Drzewo gry



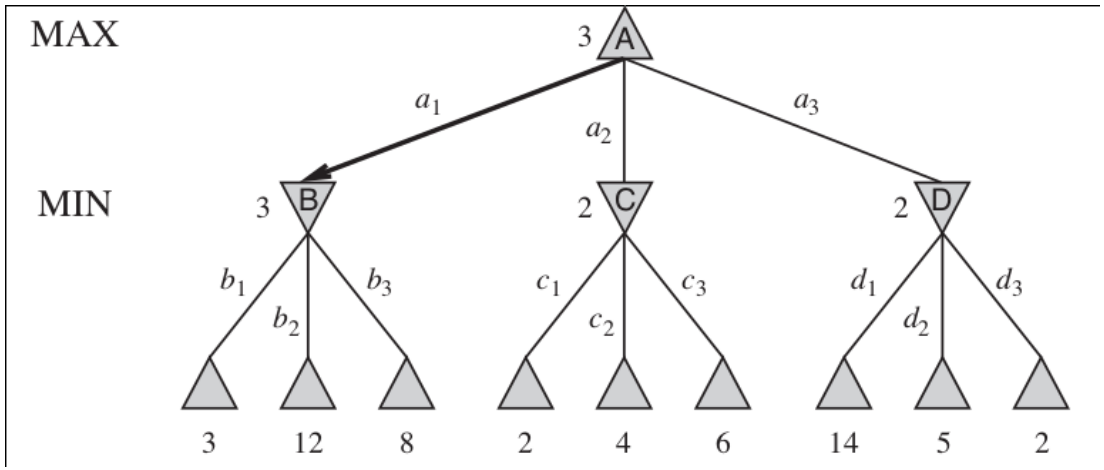
Kółko i krzyżyk. Drzewo gry

Fragment drzewa dla prawdziwej gry



Inna prosta gra (2)

- Mamy dwóch graczy Max i Min (jeden chce maksymalizacji, drugi minimalizacji).
- Wartość dla Max-a to liczba przeciwna wartości dla Min-a.
- Mamy dwa ruchy, zaczyna gracz maksymalizujący.



Algorytm MiniMax

```
MAX = 1
MIN = 0

def decision(state):
    """decision for MAX"""
    return max(a for actions(state),
       key = lambda a : minmax(result(a,state), MIN))

def minmax(state, player):
    if terminal(state): return utility(state)

    values = [minmax(result(a,state), 1-player) for a in actions(state)]
    if player == MIN:
        return min(values)
    else:
        return max(values)
```

Drobna uwaga nazewnicza

Spotyka się różne warianty nazewnicze (niestety również na naszych slajdach):

- Algorytm MiniMax
- Algorytm min-max
- Algorytm MinMax

Algorytm MiniMax

- O(d) – pamięć
- $O(b^{2d})$ czas, gdzie d jest liczbą ply's (półruchów)
- Dla szachów $b \approx 35$, $d \approx 50$
- Dla go: 250, 150

Algorytm MiniMax (wersja realistyczna)

- Algorytm MiniMax działa jedynie dla bardzo małych, sztucznych gier (ewentualnie dla końcówek prawdziwych gier).
- Żeby go uczynić realistycznym, musimy:
  a) Przerwać poszukiwania na jakiejś głębokości.
  b) Umieć szacować wartość nieterminalnych sytuacji na planszy.

Algorytm MinMax z głębokością

```
def decision(state):
    return max(a for actions(state),
       key = lambda a : minmax(result(a,state), MIN ,0))

def minmax(state, player, depth):
    if terminal(state): return utility(state)
    if cut_off_test(state, depth):
        return heuristic_value(state)

    values = [minmax(result(a,state), 1-player, depth+1) for a in actions(state)]
    if player == 0:
        return min(values)
    else:
        return max(values)
```

Algorytm MinMax z głębokością

```
def decision(state):
    return max(a for actions(state),
       key = lambda a : minmax(result(a,state), MIN ,0))

def minmax(state, player, depth):
    if terminal(state): return utility(state)
    if cut_off_test(state, depth):
        return heuristic_value(state)

    values = [minmax(result(a,state), 1-player, depth+1) for a in actions(state)]
    if player == 0:
        return min(values)
    else:
        return max(values)
```

Dwa parametry algorytmu wyszukiwania

1. cut_off_test: kiedy kończymy przeszukiwanie
   - najłatwiej: jak osiągniemy maksymalny poziom, biorąc pod uwagę możliwości
   - Nie jest to jedyne wyjście (ani najlepsze)
2. Co to znaczy funkcja heuristic_value

Jak szacować wartość sytuacji?

Wariant 1

Korzystamy z wiedzy eksperta, próbując ją sformalizować.

Wariant 2

Próbujemy zaprząc jakiś mechanizm uczenia (lub przeszukiwania), żeby tę funkcję wybrać.

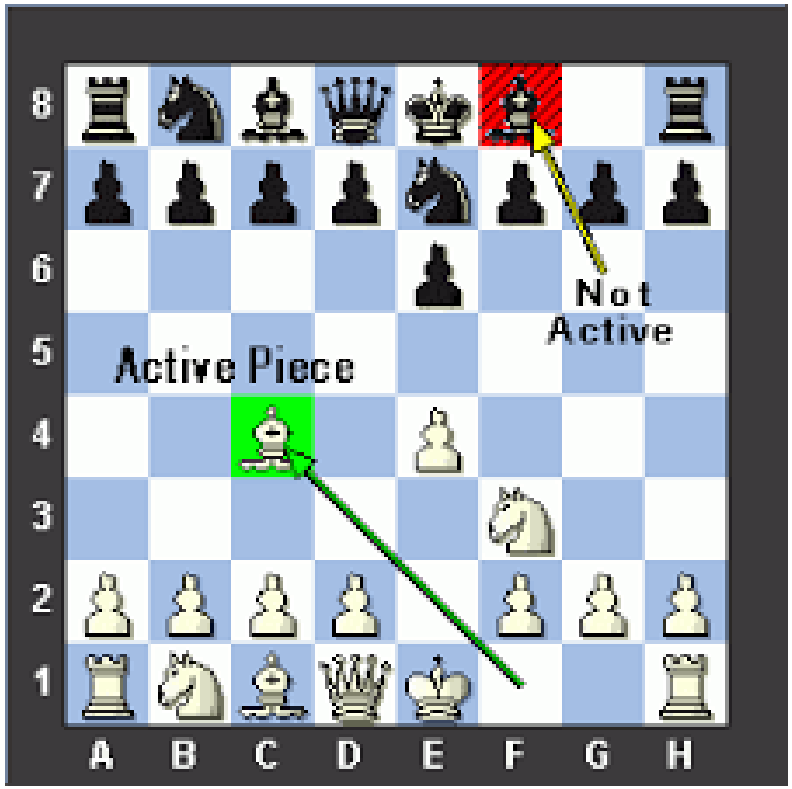
Jak szacować wartość sytuacji? (2)

Generalne wskazówki:

1. Przewaga materialna (więcej, lepszych figur)
2. Ustawienie figur (ruchliwość – liczba możliwych ruchów)
3. Szacowana liczba ruchów do zwycięstwa (zagrożony król, itp).
4. Ochrona naszych figur (jak mnie zbijesz, to ja cię zaraz zbiję)

Aktywny goniec

Biały goniec wprowadzony do gry, czarny nie może nic zrobić.



Przewaga materialna

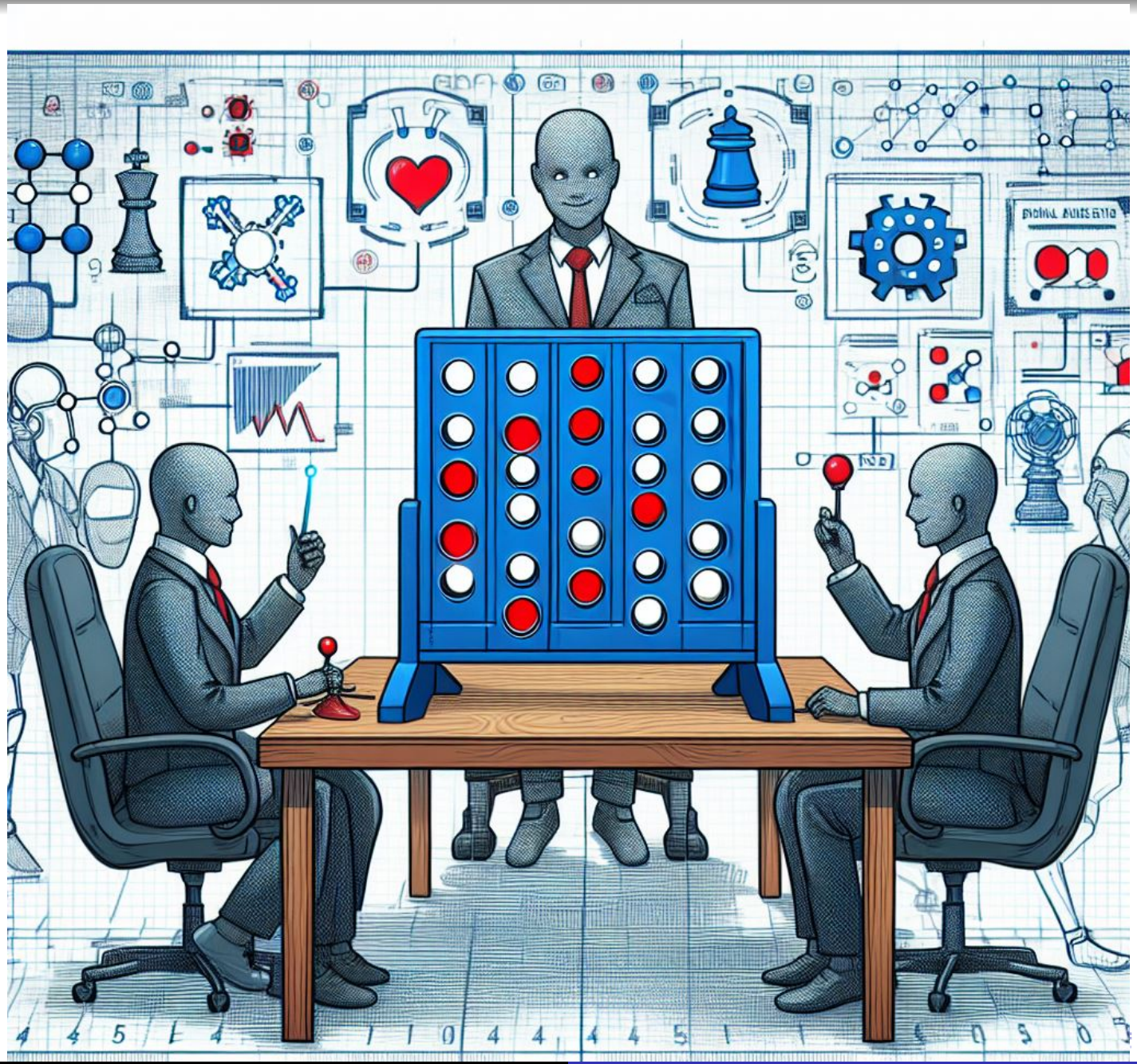
- Wartość materialną liczą powszechnie szachiści:
  a) pion: 1
  b) skoczek, goniec: 3
  c) wieża: 5
  d) hetman: 9
- Sprawdzono doświadczalnie, że te wartości są dobrze dobrane (jak sobie wyobrazić taki eksperyment?)

Uwaga

Nawet nie wiedząc nic o uczeniu, możemy sobie wyobrazić łatwo jakąś procedurę wyznaczania tych wartości. Na przykład:

1. Losujemy 100 zestawów: (1, wartość-gońca, wartość-skoczka, wartość-wieży, wartość-hetmana).
2. Przeprowadzamy pojedynki każdy z każdym.
3. Wybieramy zwycięzcę.

Connect 4. Przykładowa gra



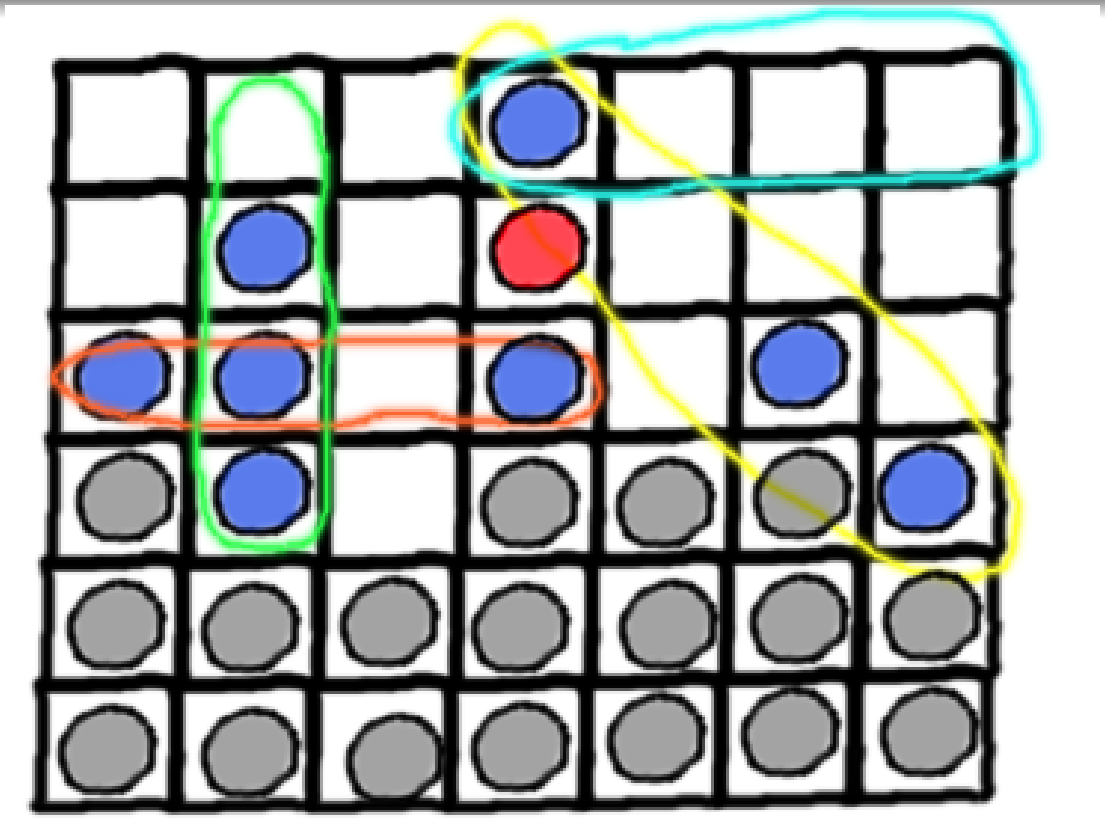
Connect 4. Przykładowa gra

- Prosty, a zarazem grywalny wariant kółka i krzyżyka
- Dodatkowe elementy: mamy ciążenie i piony spadają, gramy do 4 w wierszu, kolumnie lub na przekątnej

Connect 4. Zwycięska konfiguracja

- Prosty, a zarazem grywalny wariant kółka i krzyżyka
- Dodatkowe elementy: mamy ciążenie i piony spadają, gramy do 4 w wierszu, kolumnie lub na przekątnej

Co to znaczy wzorzec w Connect 4?



- Analizujemy wszystkie czwórki pól (w każdej bowiem może się zdarzyć układ wygrywający)
- Czwórki, w których są pionki obu kolorów pomijamy
- Wyznaczamy wagę 1-ek, dwójek, trójek (być może zależnie od kierunków)

Wzorce w Connect 4

- Możliwe są większe wzorce, uwzględniające szerszy kontekst
- możliwe jest również **uczenie** większych wzorców. Na przykład za pomocą splotowych sieci neuronowych (CNN).

Uwaga

Takie sieci działały w AlphaGo.

Ewaluacja z wzorców

- Funkcja oceny może być ważoną sumą zaobserwowanych wzorców.
- Wzór:

$$\sum_i w_i p_i$$

(w_i – waga i-tego wzorca, p_i – ile razy ten wzorzec występuje na planszy)

- Niektóre wagi są dodatnie (mój dobry wzorzec, słabe ustawienie oponenta), inne ujemne.

Drobna uwaga o ewolucji w grach: jak wyznaczyć parametry funkcji oceniającej?

- Istnieje pokusa, żeby zastosować algorytmy ewolucyjne (bo zadanie przypomina ewolucje, w której osobniki toczą ze sobą walkę).
- **Problem:** Jak wyznaczyć funkcję celu?
  a) Rozgrywać turnieje, przystosowaniem jest średni wynik.
  b) Wybrać grupę przeciwników (stałą), przystosowaniem X-a będzie średni wynik z tymi przeciwnikami.

Uwaga

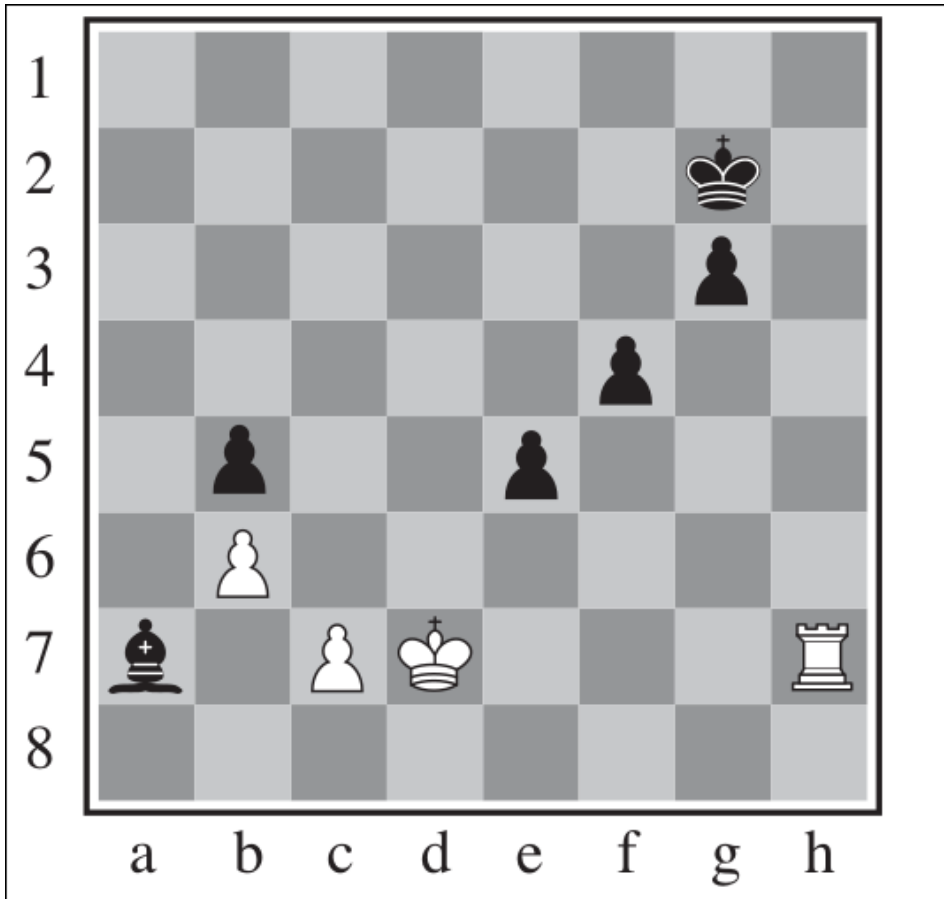
Opcja pełnej ewolucji trochę niebezpieczna, często łączy się oba warianty.

Przerywanie przeszukiwania

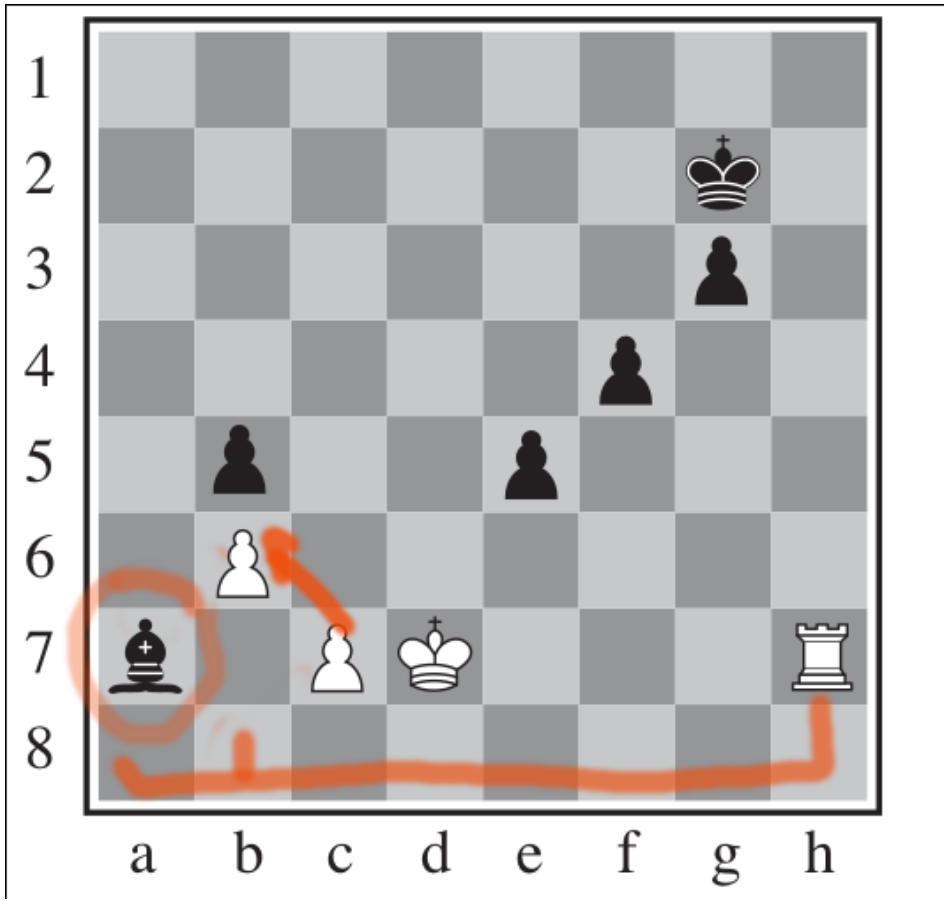
Drugi metaparametr funkcji obliczającej wartość planszy.

- Są dwa problemy związane z przerywaniem przeszukiwania:
  1. Przerwanie w niestabilnej sytuacji (na przykład w środku wymiany hetmanów)
  2. Tzw. efekt horyzontu (czyli widzimy, że coś się zdarzy, ale w odległej perspektywie)

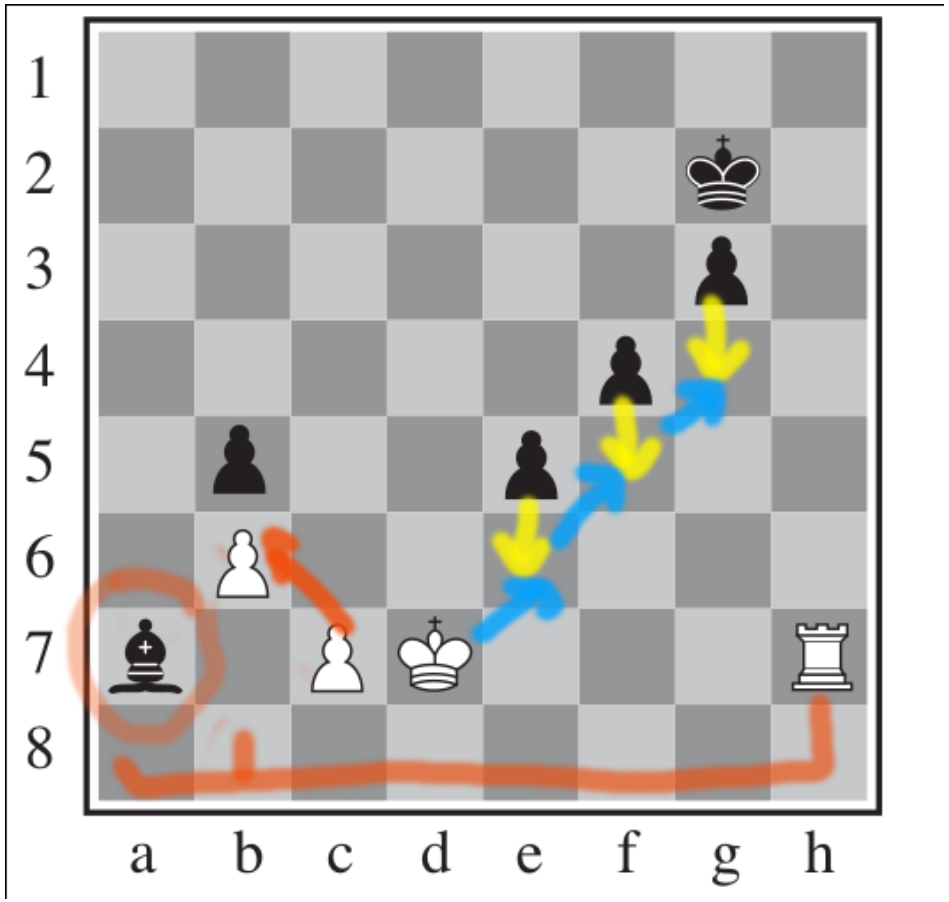
Efekt horyzontu (zła sytuacja czarnego gońca)



Efekt horyzontu (zła sytuacja czarnego gońca)



Efekt horyzontu (zła sytuacja czarnego gońca)



Kończenie przeszukiwań w praktyce

- Nieprzerywanie, jeżeli przeciwnik ma bicie.
- Ogólniej: powyżej jakiejś głębokości rozważamy tylko ruchy mocno zmieniające sytuację

Definicja

W **przeszukiwaniu z bezruchem** (quiescence search) możemy skończyć poszukiwanie **tylko** gdy sytuacja jest statyczna.

Kończenie przeszukiwań w praktyce

- Można też stosować jakąś wersję *local beam search* (od któregoś momentu ograniczając mocno rozgałęzienie drzewa)
- Rozważa się warunek singular extension, czyli istnienie jednego ruchu, który jest wyraźnie (na oko) lepszy od innych. Takie ruchy zawsze wykonujemy, zwiększając głębokość, a nie zwiększając rozgałęzienia.

Uwaga

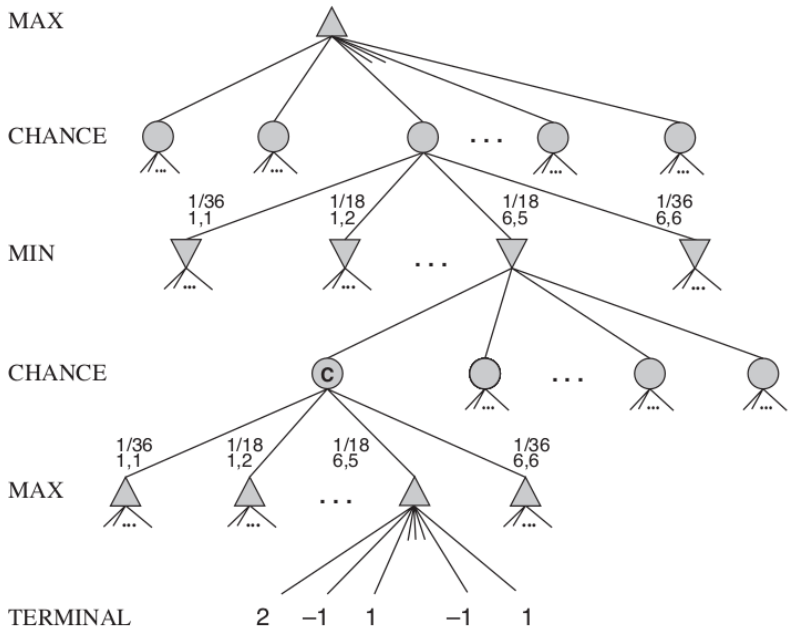
Trochę tak działają ludzie.

Losowość w grach

- W niektórych grach (i w życiu) mamy element losowy.
- Prosty przykład: szachy z kostką:
  - Przed ruchem wykonujemy rzut kostką, który determinuje czym możemy się ruszyć,
  - 1 -pionek, 2 - skoczek, 3 - goniec, 4 – wieża, 5 – hetman, 6 – król
  - Gramy do zbicia króla.

Losowość w grach

- Wprowadzamy dodatkowe węzły, czyli chance nodes.
- Przykładowe drzewo gry (dla losowania przy użyciu dwóch kości):



Expectimax

- Minimax, do którego dołożono węzły losowe.
- W węzłach losowych mamy wybór wartości oczekiwanej (sumowanie)

```
def emm(state, player):
    if terminal(state): return utility(state)
    if player == MIN:
        return min( emm(result(state, a), next(player)) for a in actions(state))
    if player == MAX:
        return max( emm(result(state, a), next(player)) for a in actions(state))
    if player == CHANCE:
        return sum( P(r) * emm(result(state, r), next(player)) for r in actions(state))
```

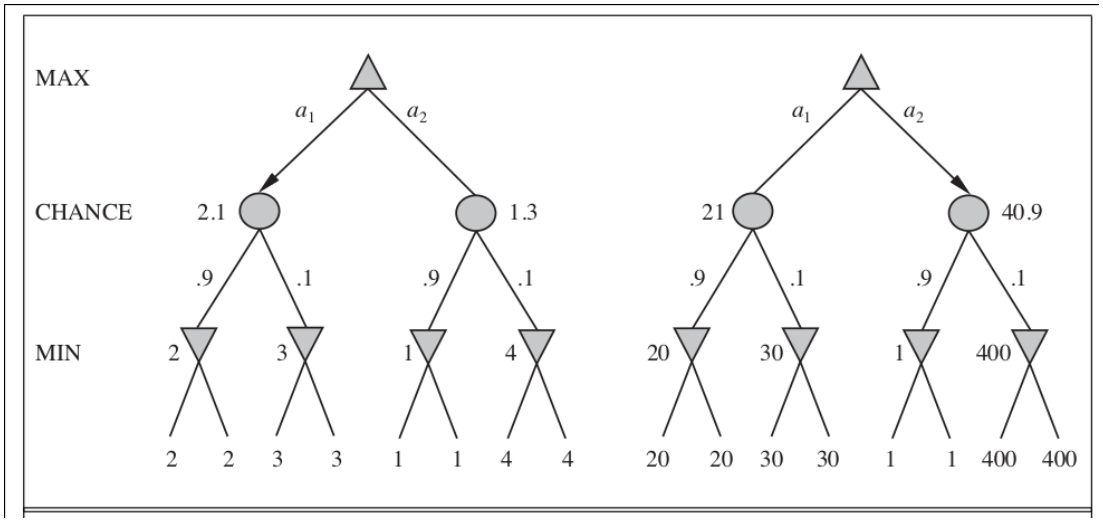
Wartość sytuacji w grach z losowością

Uwaga 1

Dowolne monotoniczne przekształcenie nie zmienia ruchów wybieranych przez minimax!

Uwaga 2

W grach z losowością powyższe zdanie przestaje być prawdziwe.



Monte Carlo Simulation

- Analiza gier z losowością jest nieco trudniejsza.
- Możemy skorzystać z następującej idei:

  Oceniamy sytuację przeprowadzając dużo losowych gier rozpoczynających się w danej sytuacji

- **Uwaga:** dwa rodzaje losowości: jeden związany z węzłami losowymi (dany przez grę), drugi związany z węzłami min/max – zamiast wyliczać ruch wykonujemy ruch losowy.

Uwaga

Monte Carlo Simulation dotyczy nie tylko gier z losowością!

Monte Carlo Simulation

- Zauważmy, że Monte Carlo Simulation jakoś rozwiązuje problem horyzontu (bo symulacje mogą być b. długie)
- Możemy losować ruchy z niejednakowym prawdopodobieństwem (preferując te, które lokalnie wyglądają sensownie)

Uwaga

Bardzo ważnym nie tylko w grach jest algorytm Monte Carlo Tree Search, o którym jeszcze powiemy.