Jeszcze o grach

Paweł Rychlikowski

Instytut Informatyki UWr

17 kwietnia 2019

O grach. Przypomnienie

1. Zadania przeszukiwania, w którym zakładamy, że istnieje oponent (w grach o sumie zerowej oponent złośliwy).
2. Do efektywnego grania w nietrywialne gry potrzebna heurystyczna funkcja oceniająca stany (na przykład licząca nasze i przeciwnika bierki i ich siłę)

Uwaga

Parametry tej funkcji również mogą być celem przeszukiwania!

Connect 4. Inna 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. Inna 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

Co to znaczy wzorzec w Connect 4?

Czyli jak napisać funckję oceniającą?

- **Podejście 1**: liczymy 1-ki, 2-ki, 3-ki i 4-ki w rzędzie, kolumnie i po przekątnej
  - Jedynki: `..o.`, `.x..`, ...
  - Dwójki: `.oo.`, `o.o.`, `x..x`, ...
  - Trójki: `ooo.`, `xx.x` ...
  - Czwórki: `oooo`, `xxxx` ...
  - Układy bezwartościowe: `o..x`, `ooox`, ... – liczymy jako zera, bo nie rozwiną się do układu wygrywającego.
- Można różnicować na kierunki.
- Można inaczej traktować `ooo.` oraz `oo.o`

Uwaga

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

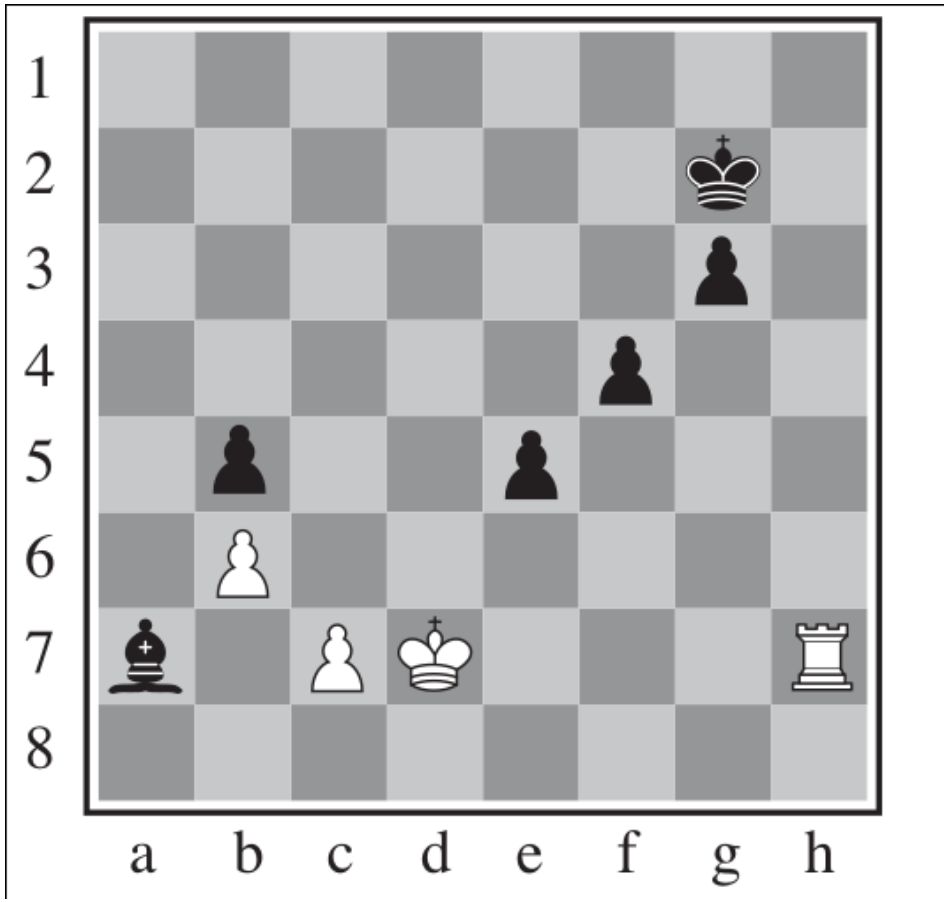
- Niektóre wagi są dodatnie (mój dobry wzorzec, słabe ustawienie oponenta), inne ujemne.

Przerywanie przeszukiwania

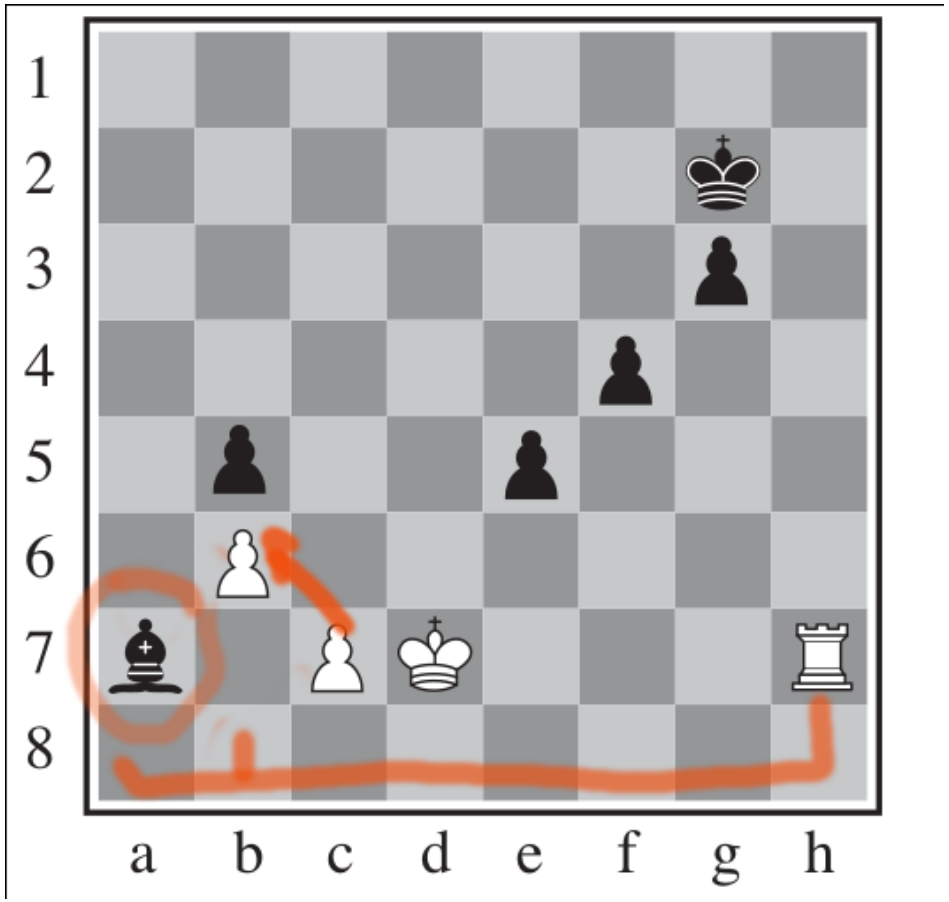
Drugi metaparametr funkcji obliczającej wartość planszy.

- 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)

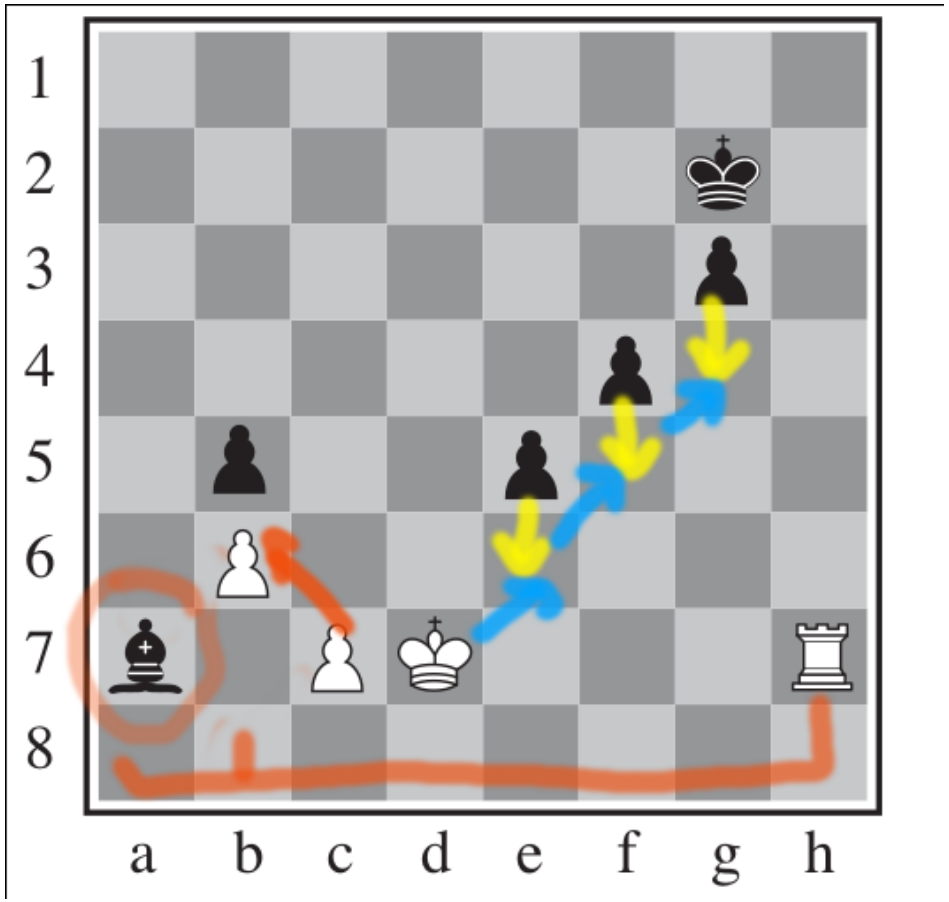
Efekt horyzontu



Efekt horyzontu



Efekt horyzontu



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.

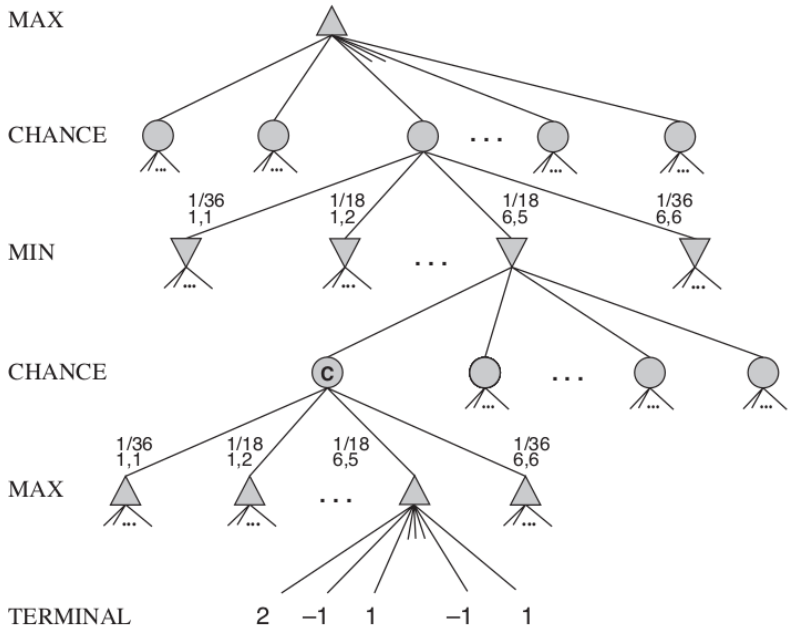
Losowość w grach

- 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.

f

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))
```

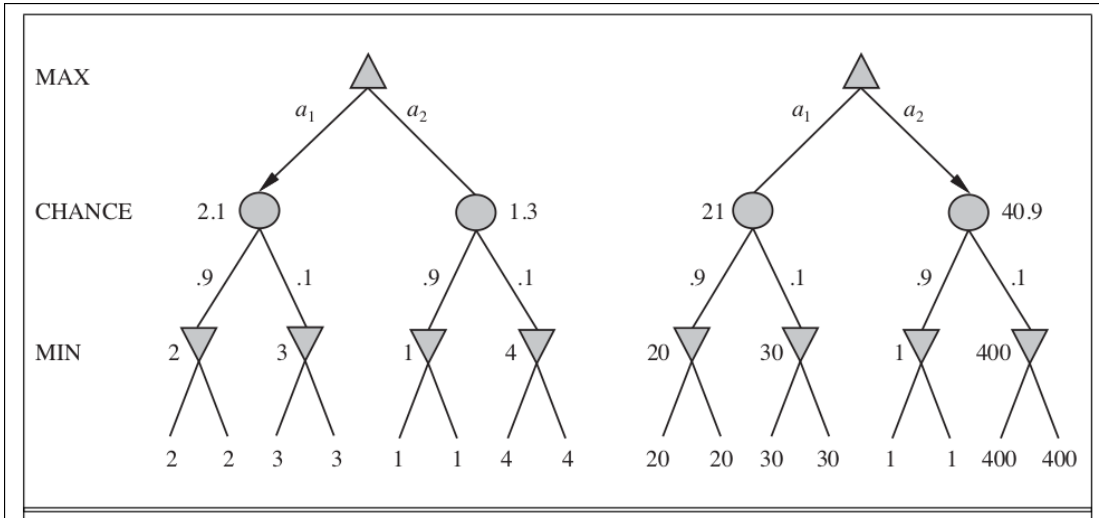
Wartość sytuacji w grach z losowością

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!

Gry częściowo obserwowalne

- Ciekawe do analizy są gry, w których agenci nie mają pełnej wiedzy o świecie.
- Klasyczne przykłady to gry karciane, ale nie tylko.

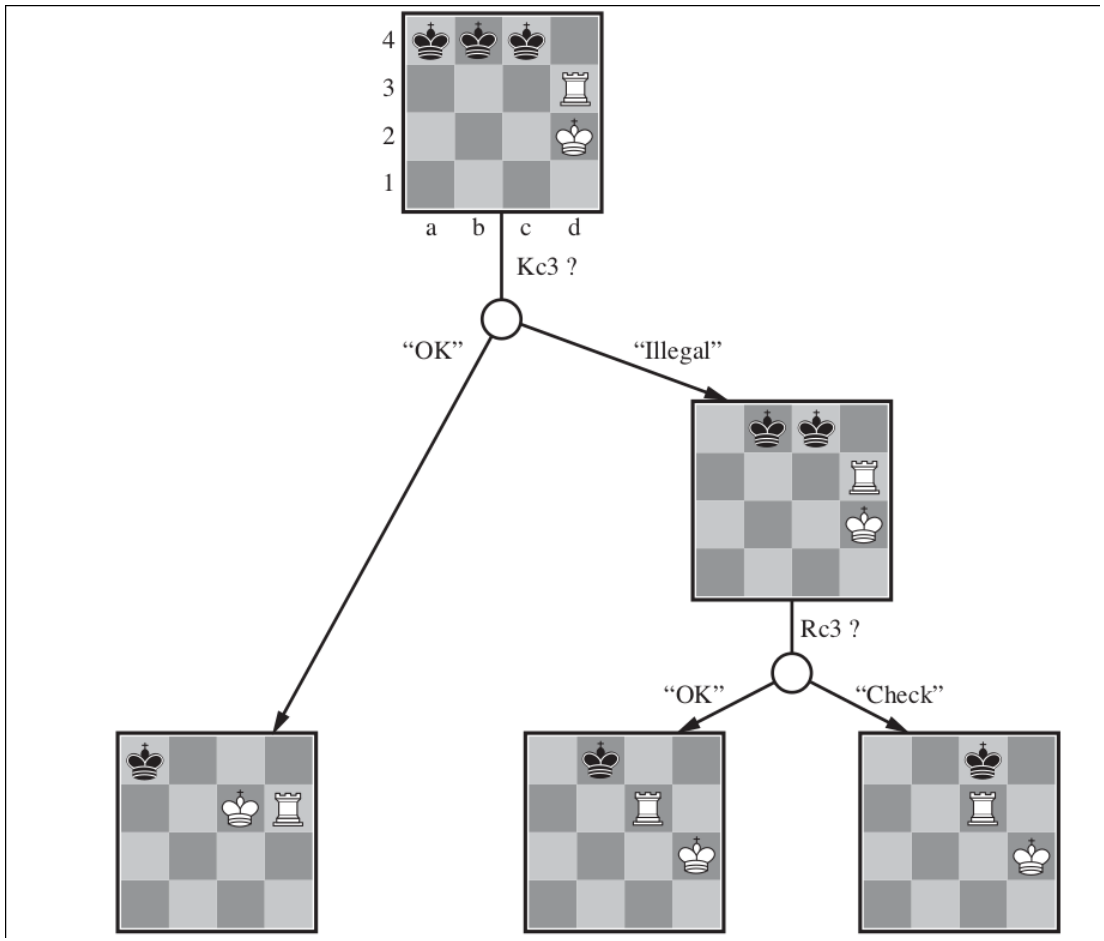
Kriegspiel

- Mamy dwóch graczy, arbitra i 3 szachownice.
- Gracze widzą na szachownicy swoje pionki, mogą tworzyć też hipotezy o bierkach przeciwnika.
- Arbiter zna położenie wszystkich figur i udziela graczom pewnych (skąpych) informacji.
  a) przede wszystkim ocenia, czy ruch jest możliwy (komunikacja osobista, dobry ruch jest od razu wykonywany, w przypadku złego, gracz proponuje kolejny, aż do skutku)
  b) odpowiada na pytanie: „czy ja (gracz) mam jakieś bicie?"
  c) informuje obu graczy, że „na polu X zbito bierkę" (nie podając jaka bierka jest zbita, a jaka biła)
  d) Mówi o szachu (do ubu graczy), dodając, że zagrożenie jest w wierszu, kolumnie, przekątnej lub przez skoczka
- Tak poza tym, to całkiem normalne szachy.

Podobno ludzie radzą sobie z tą grą całkiem nieźle...

Końcówka w Kriegspiel

Przykładowa końcówka, gracz biały dowiedział się, że czarnemu został tylko król i jest on na jednym z 3 pól.



Uwaga 1

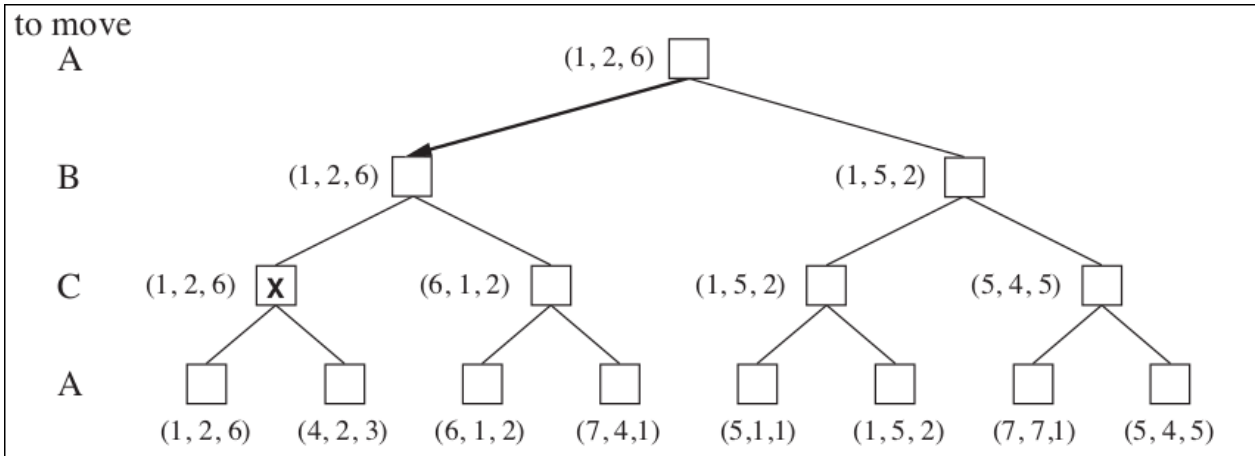
W stanie gry powinniśmy umieścić możliwe ustawienia bierek przeciwnika

Trochę jak z komandosem...

Pytanie

A jak grać w brydża, bądź inną grę karcianą?

Gry z większą liczbą uczestników



Gry wieloosobowe. Problemy

- Strategia maksymalizująca korzyść pojedynczego gracza w oczywisty sposób nieoptymalna (A mógłby się dogadać z B).
- Kwestie sojuszów, zrywania sojuszów, budowania wiarygodności.
- Czasem używa się: paranoidalnego założenia – gra wieloosobowa staje się jednoosobową, w której oni wszyscy chcą mi zaszkodzić.

Księga otwarć

Uwaga

Początki gier są podobne (bo rozpoczynamy z tego samego stanu startowego)

Z tego wynika, że:

- Możemy np. poświęcić parę godzin, na obliczenie najlepszej odpowiedzi na każdy ruch otwierający.
- Możemy „rozwinąć" początkowy kawałek drzewa (od któregoś momentu tylko dobre odpowiedzi oponenta)
- Możemy skorzystać z literatury dotyczącej początków gry (obrona sycylijska, partia katalońska, obrona bałtycka, i wiele innych)

Spamiętywanie

- Stany mogą się powtarzać (również z zeszłej partii naszego programu).
- Jeżeli mamy oceniony stan z głębokością 6 i dochodzimy do niego z głębokością 3, to opłaca się wziąć tę bardziej prezycyjną ocenę (w dodatku bez żadnych obliczeń).

Uwaga

Potrzebny nam jest efektywny sposób pamiętania sytuacji na planszy.

Tabele transpozycji

- Zapamiętywanie pozycji powinno być efektywne pamięciowo i czasowo.
- Używa się następującego schematu kodowania (Zobrist hashing:
  - Mamy zdania typu: biały goniec jest na g6 (WB-G6), czarny król jest na b4 (BK-B4), itd (12×64)
  - Każde z nich dostaje losowy ciąg bitów (popularny wybór: 64 bity)
  - Planszę kodujemy jako **xor** wszystkich prawdziwych zdań o tej planszy.
  - Zauważmy, jak łatwo przekształca się te kody:
    stary-kod = stary-kod **xor** wk-a4 **xor** wk-b5
    to ruch białego króla z a4 na b5

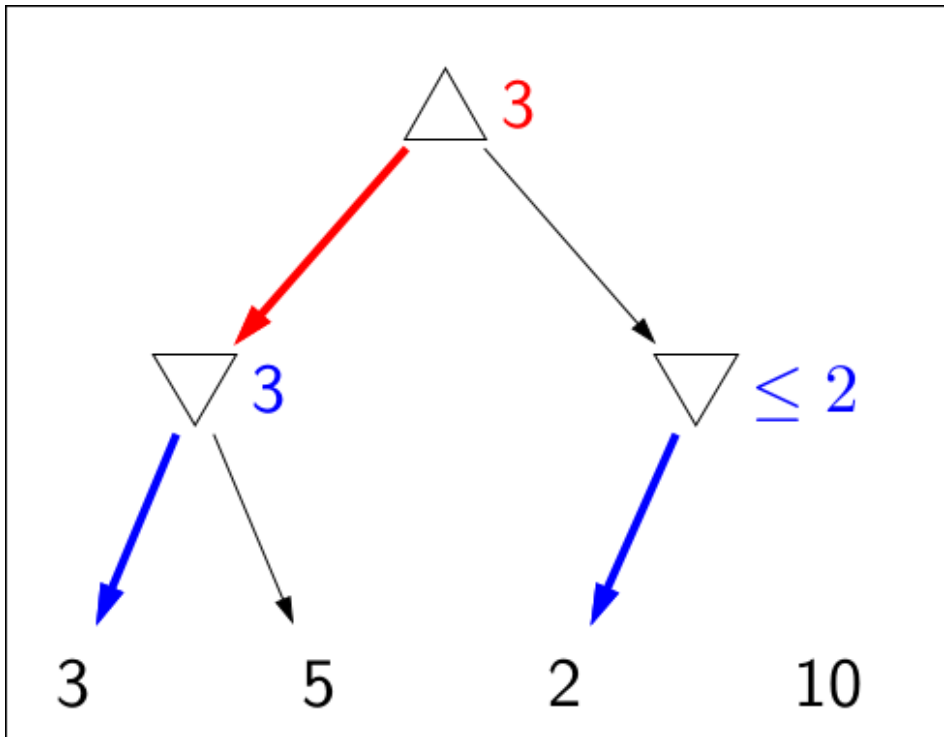
Uwaga

Często nie przejmujemy się konfliktami, uznając że nie wpływają w znaczący sposób na rozgrywkę.

Algorytm Alpha-Beta Search

- Idea 1: nie zawsze musimy przeglądać całe drzewo, żeby wybrać optymalną ścieżkę.
- Idea 2: mamy dwa ruchy
  1. Pierwszy ma wartość z przedziału [2,**5**]
  2. Drugi ma wartość z przedziału [**5**,100]
- Nie musimy ustalać dokładnej wartości pierwszego ruchu, by stwierdzić, że drugi jest lepszy dla Maxa!

Obcinanie fragmentów drzew



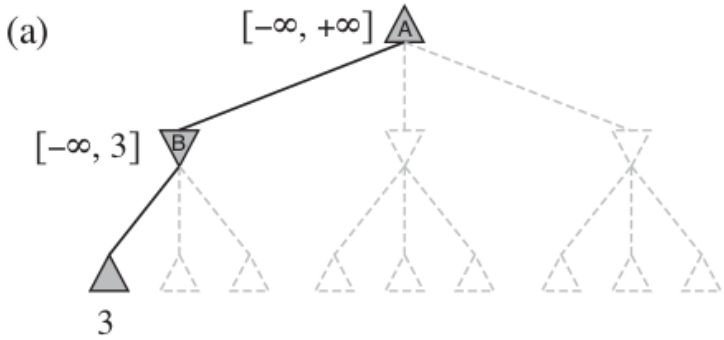
Źródło: CS221, Liang i Ermon

Mamy: $\max(3, \leq 2) = 3$

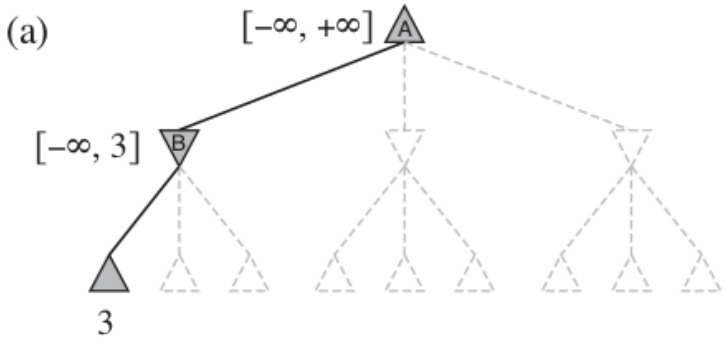
Obcinanie fragmentów drzew

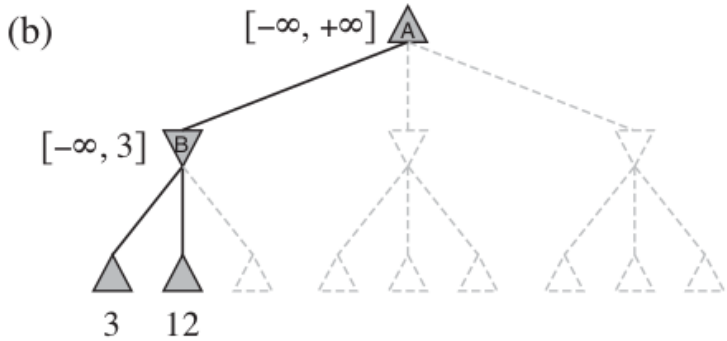
- Jeżeli możemy udowodnić, że w jakimś poddrzewie nie ma optymalnej wartości, to możemy pominąć to poddrzewo.
- Będziemy pamiętać:
  - α – dolne ograniczenie dla węzłów MAX ($\geq \alpha$)
  - β – górne ograniczenie dla węzłow MIN ($\leq \beta$)

Alfa-Beta w akcji

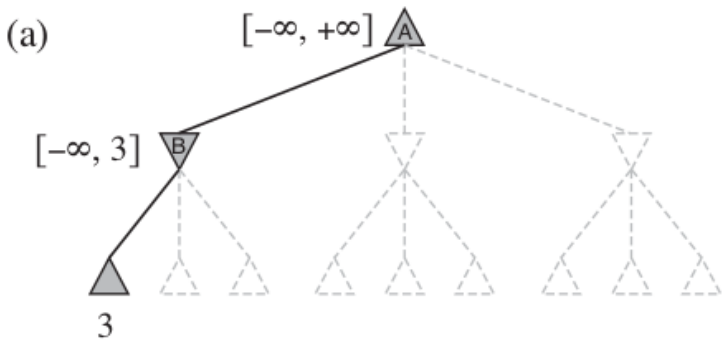


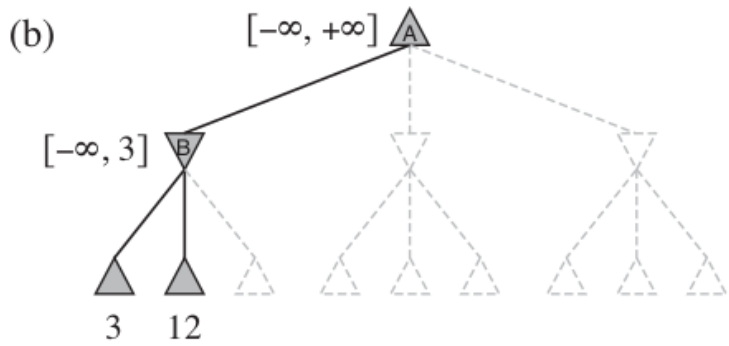
Alfa-Beta w akcji

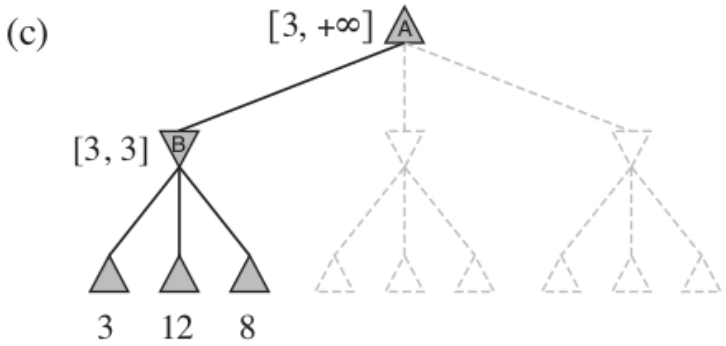




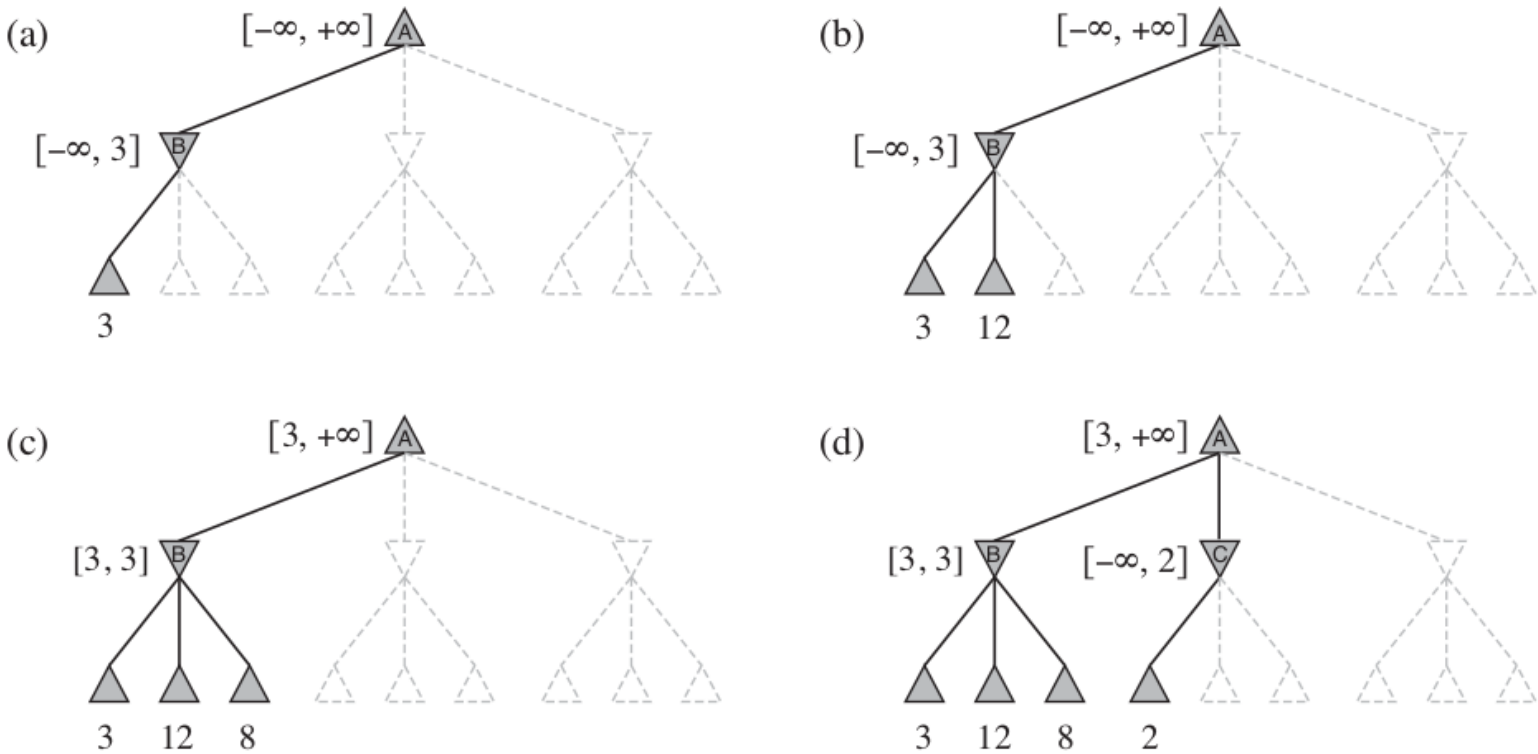
Alfa-Beta w akcji



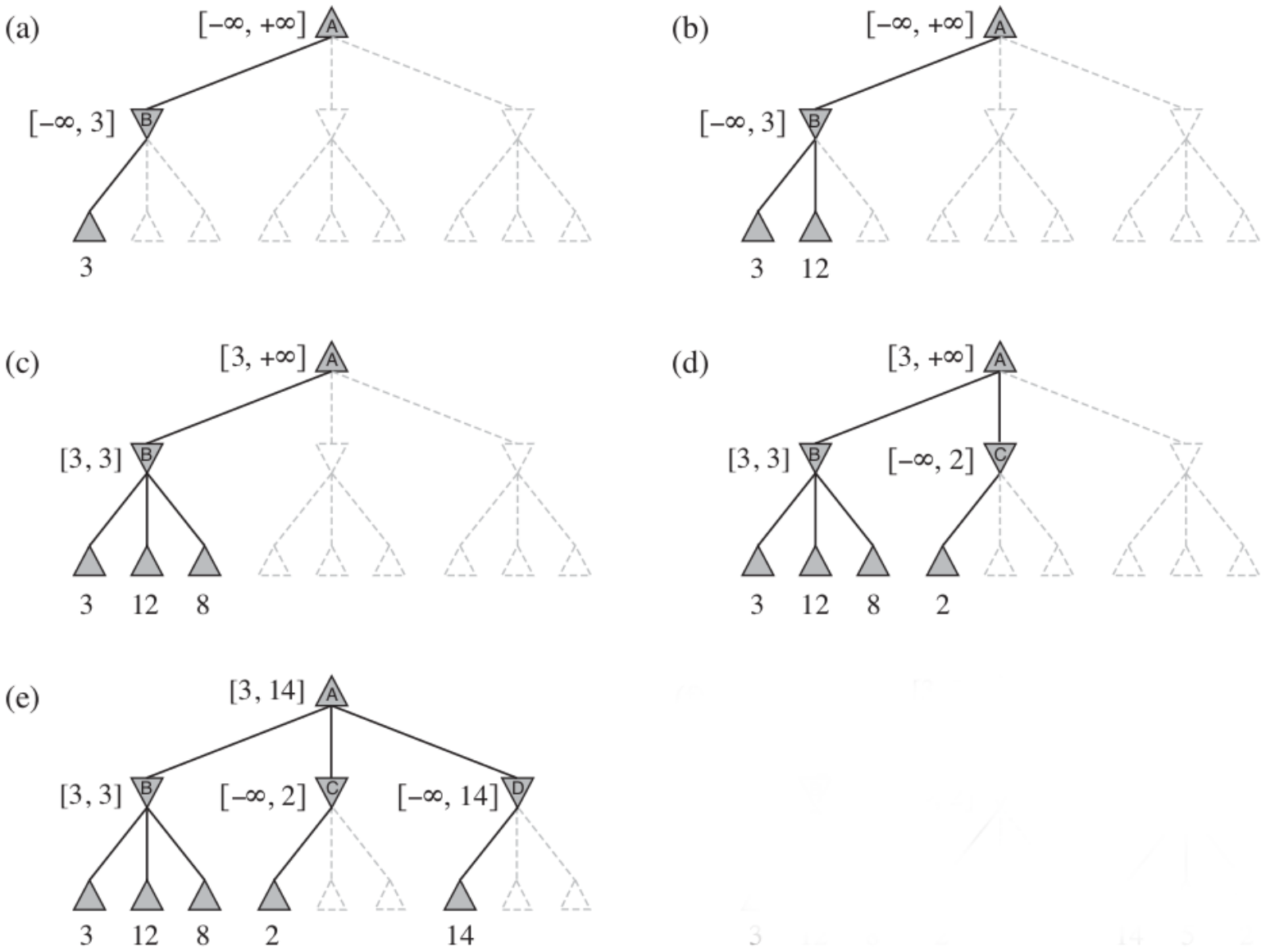




Alfa-Beta w akcji



Alfa-Beta w akcji



Alfa-Beta w akcji