Sztuczna Inteligencja i Systemy Doradcze

Przeszukiwanie przestrzeni stanów — gry

Gry a problemy przeszukiwania

"Nieprzewidywalny" przeciwnik $\Rightarrow$ rozwiązanie jest strategią
specyfikującą posunięcie dla każdej możliwej odpowiedzi przeciwnika

Ograniczenia czasowe $\Rightarrow$ Mało prawdopodobne znalezienie celu, trzeba aproksymować

Historia:

- Komputer rozważa różne scenariusze rozgrywki (Babbage, 1846)
- Algorytmy dla gier z pełną inform. (Zermelo, 1912; Von Neumann, 1944)
- Skończony horyzont, aproksymacyjna ocena stanu gry (Zuse, 1945; Wiener, 1948; Shannon, 1950)
- Pierwszy program grający w szachy (Turing, 1951)
- Zastosowanie uczenia maszynowego do poprawy trafności oceny stanu gry (Samuel, 1952–57)
- Odcięcia umożliwiające głębsze przeszukiwanie (McCarthy, 1956)

Rodzaje gier

	deterministyczne	**niedeterministyczne**
Pełna informacja	szachy, warcaby, go, otello	backgammon, monopoly
Niepełna informacja		bridge, poker, scrabble, nuclear war

Gra deterministyczna: 2 graczy

Gracze: **MAX** i **MIN**

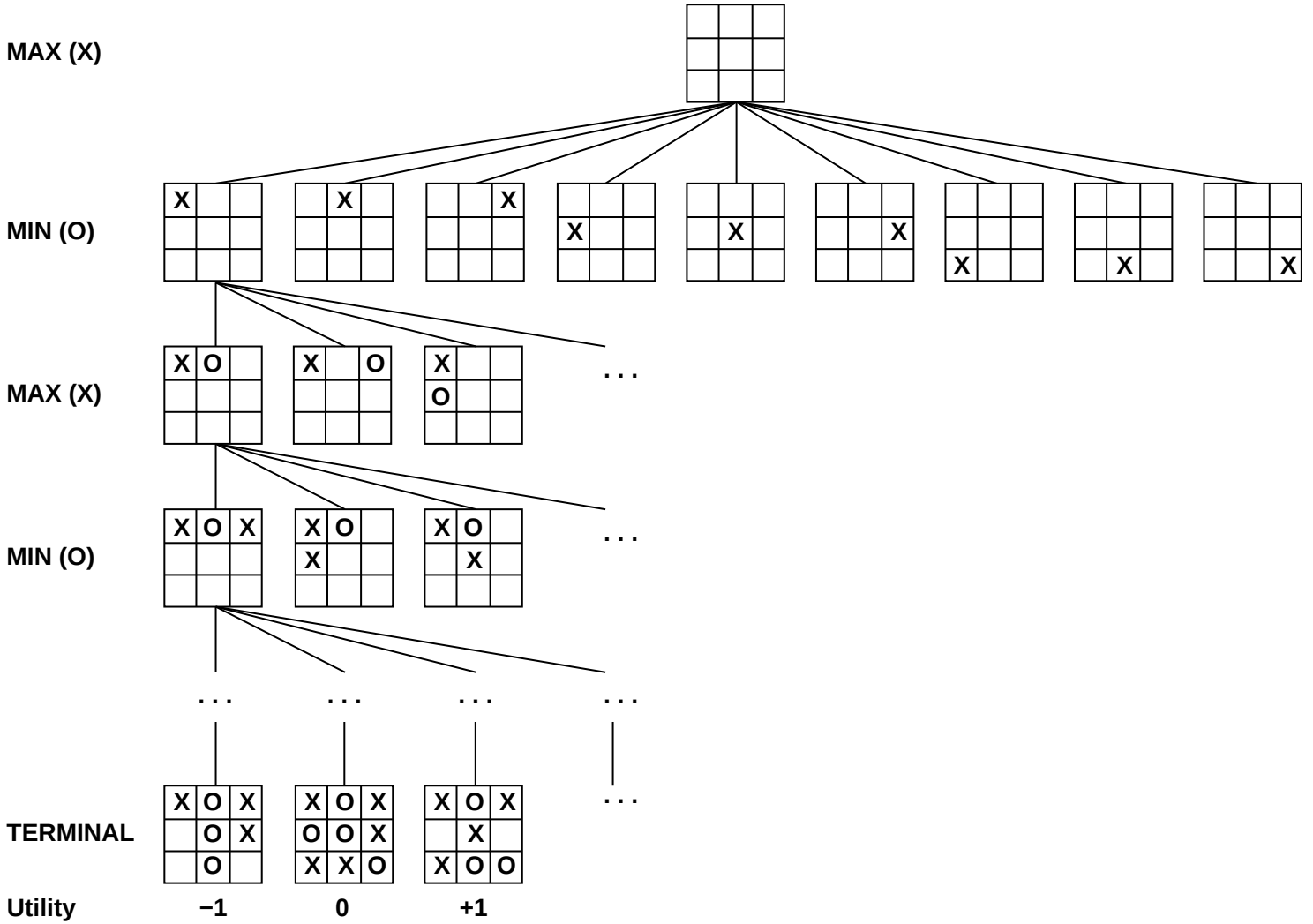
Stan początkowy: stan planszy i wskazanie gracza rozpoczynającego (**MAX**)

Funkcja następnika: zbiór par $(posunięcie, stan)$ opisujących wszystkie dopuszczalne posunięcia z bieżącego stanu

Test końca gry: sprawdza, czy stan gry jest końcowy

Funkcja użyteczności (wypłaty): numeryczna wartość dla stanów końcowych
np. wypłaty dla wygranej, porażki i remisu
mogą być odpowiednio +1, -1 i 0.

Drzewo gry deterministycznej: 2 graczy



Strategia minimax: algorytm

Dla gier deterministycznych z pełną informacją

Pomysł: wybiera ruch zapewniający największą wypłatę

tzn. największą *wartość minimax* (funkcja MINIMAX-VALUE)

przy założeniu, że przeciwnik gra optymalnie

function MINIMAX-DECISION(*state*, *game*) **returns** *an action*

action, *state* ← the *a*, *s* **in** SUCCESSORS(*state*)
such that MINIMAX-VALUE(*s*, *game*) is maximized

return *action*

---

function MINIMAX-VALUE(*state*, *game*) **returns** *a utility value*

if TERMINAL-TEST(*state*) **then**
return UTILITY(*state*)
else if MAX is to move in *state* **then**
return the highest MINIMAX-VALUE of SUCCESSORS(*state*)
else
return the lowest MINIMAX-VALUE of SUCCESSORS(*state*)

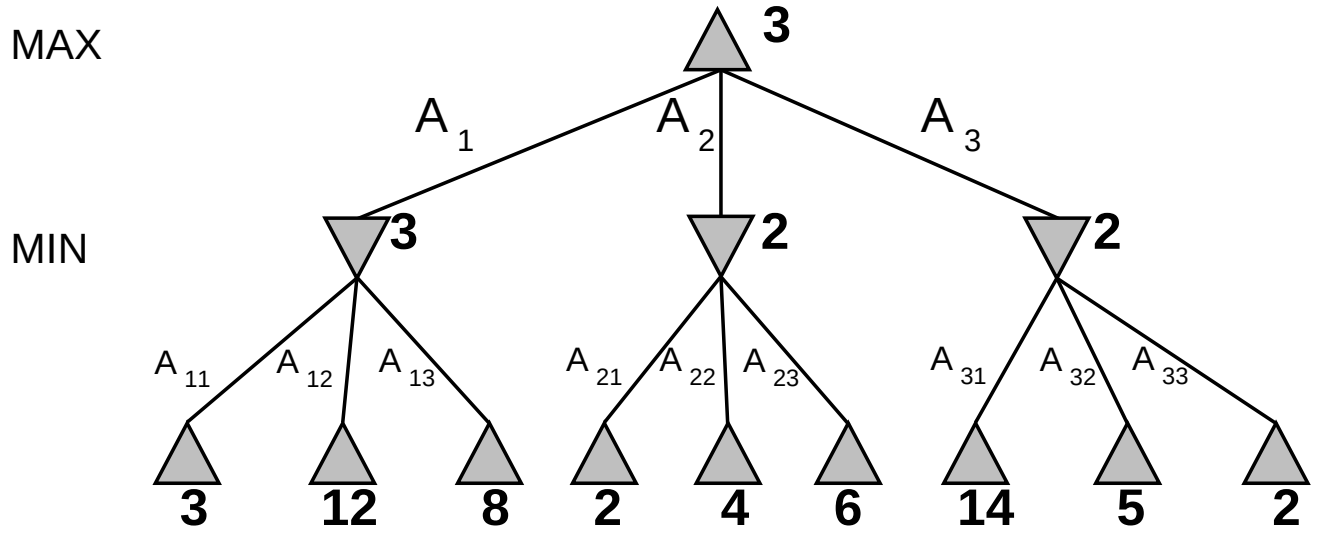
Strategia minimax: przyklad

Gracz **MAX** maksymalizuje funkcję wypłaty (węzły $\triangle$)

$\Rightarrow$ wybiera ruch w lewą gałąź drzewa

Gracz **MIN** minimalizuje funkcję wypłaty (węzły $\triangledown$)

$\Rightarrow$ wybiera ruch do lewego liścia poddrzewa



Strategia minimax: wlasnosci

Użyteczność??

Strategia minimax: wlasnosci

Użyteczność?? Gry determinist. z pełną informacją z dowolną liczbą graczy

Pełność??

Strategia minimax: wlasnosci

Użyteczność?? Gry determinist. z pełną informacją z dowolną liczbą graczy

Pełność?? Tak, jeśli drzewo przeszukiwań jest skończone
Gry z nieskończonym drzewem przesz. mogą mieć strategie skończone!

Optymalność??

Strategia minimax: wlasnosci

Użyteczność?? Gry determinist. z pełną informacją z dowolną liczbą graczy

Pełność?? Tak, jeśli drzewo przeszukiwań jest skończone

Gry z nieskończonym drzewem przesz. mogą mieć strategie skończone!

Optymalność?? Tak, jeśli przeciwnik gra optymalnie

W ogólności nieoptymalne

Złożoność czasowa??

Strategia minimax: wlasnosci

Użyteczność?? Gry determinist. z pełną informacją z dowolną liczbą graczy

Pełność?? Tak, jeśli drzewo przeszukiwań jest skończone

Gry z nieskończonym drzewem przesz. mogą mieć strategie skończone!

Optymalność?? Tak, jeśli przeciwnik gra optymalnie

W ogólności nieoptymalne

Złożoność czasowa?? $O(b^m)$

Złożoność pamięciowa??

Strategia minimax: wlasnosci

Użyteczność?? Gry determinist. z pełną informacją z dowolną liczbą graczy

Pełność?? Tak, jeśli drzewo przeszukiwań jest skończone

Gry z nieskończonym drzewem przesz. mogą mieć strategie skończone!

Optymalność?? Tak, jeśli przeciwnik gra optymalnie

W ogólności nieoptymalne

Złożoność czasowa?? $O(b^m)$

Złożoność pamięciowa?? $O(bm)$ (przezzukiwanie wgłąb)

Strategia minimax: wlasnosci

Użyteczność?? Gry determinist. z pełną informacją z dowolną liczbą graczy

Pełność?? Tak, jeśli drzewo przeszukiwań jest skończone
Gry z nieskończonym drzewem przesz. mogą mieć strategie skończone!

Optymalność?? Tak, jeśli przeciwnik gra optymalnie
W ogólności nieoptymalne

Złożoność czasowa?? $O(b^m)$

Złożoność pamięciowa?? $O(bm)$ (przezzukiwanie wgłąb)

Dla szachów, $b \approx 35$, $m \approx 100$ dla "sensownych" rozgrywek
$\Rightarrow$ dokładne rozwiązanie zupełnie nieosiągalne

Strategia minimax z odcieciem

Problem:

brak czasu na pełne przeszukanie przestrzeni stanów
np. 100 sekund na posunięcie, szybkość 10^4 węzłów/sek
$\Rightarrow 10^6$ węzłów na ruch

Rozwiązanie:

przeszukiwanie z odcięciem ograniczającym głębokość przeszukiwania

Strategia minimax z odcieciem: algorytm

function MINIMAX-DECISION(*state*, *game*) **returns** *an action*

action, *state* ← the *a*, *s* **in** SUCCESSORS(*state*)
such that MINIMAX-CUTOFF(*s*, *game*) is maximized

return *action*

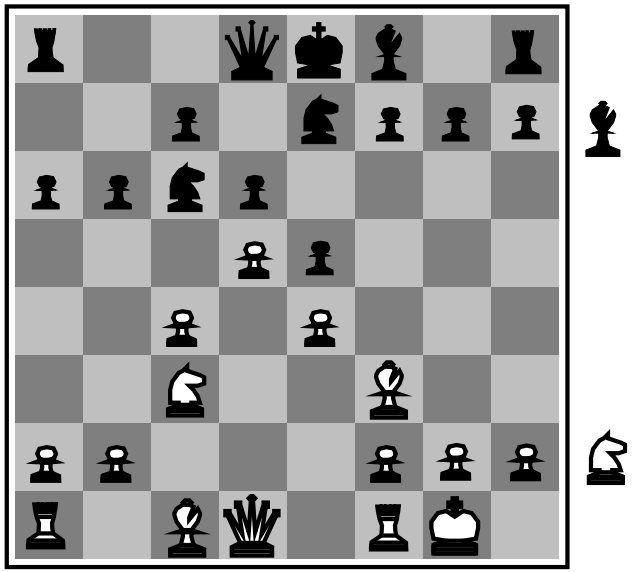
---

function MINIMAX-CUTOFF(*state*, *game*) **returns** *a utility value*

if CUTOFF-TEST(*state*) **then**
return EVAL(*state*)
else if MAX is to move in *state* **then**
return the highest MINIMAX-VALUE of SUCCESSORS(*state*)
else
return the lowest MINIMAX-VALUE of SUCCESSORS(*state*)

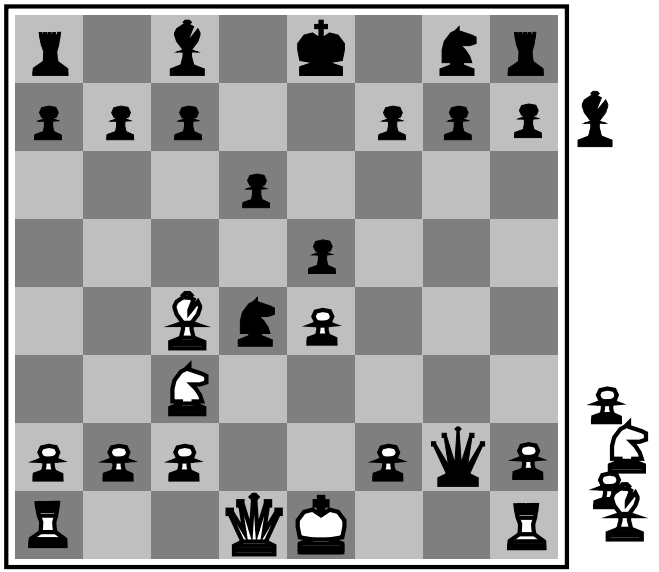
Funkcja oceny EVAL szacuje wypłatę dla danego stanu gry
= rzeczywistej wypłacie dla stanów końcowych

Funkcja oceny: przyklad



Black to move

White slightly better



White to move

Black winning

Dla szachów, przeważnie *liniowa* ważona suma cech

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

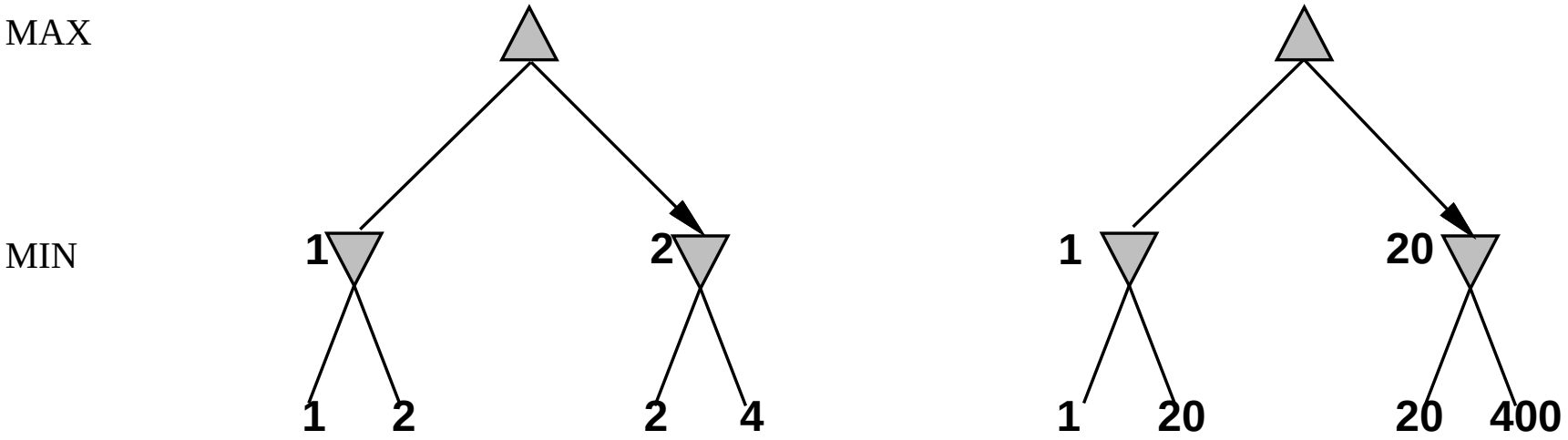
np. $w_1 = 9$ z

$f_1(s) =$ (liczba białych hetmanów) – (liczba czarnych hetmanów), itd.

Strategia minimax z odcieciem: wlasnosci

Funkcja oceny wypłaty EVAL w grach deterministycznych ma znaczenie wyłącznie *porządkujące*

$\Rightarrow$ zachowuje działanie przy dowolnym przekształceniu *monotonicznym* funkcji EVAL



Strategia minimax z odcieciem: skutecznosc

W praktyce dla szachów

$$b^m = 10^6, \quad b = 35 \quad \Rightarrow \quad m = 4$$

4-warstwowe przeszukiwanie $\approx$ nowicjusz

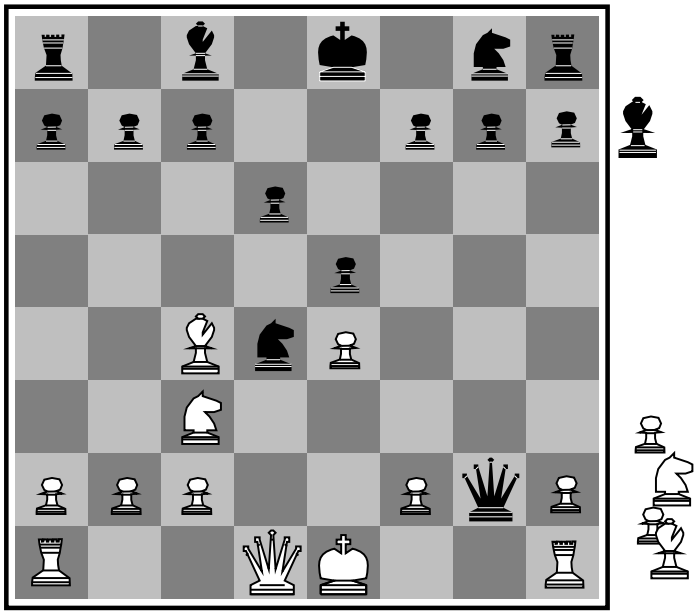
Potrzeba lepiej:

8-warstwowe przeszukiwanie $\approx$ typowy PC, mistrz

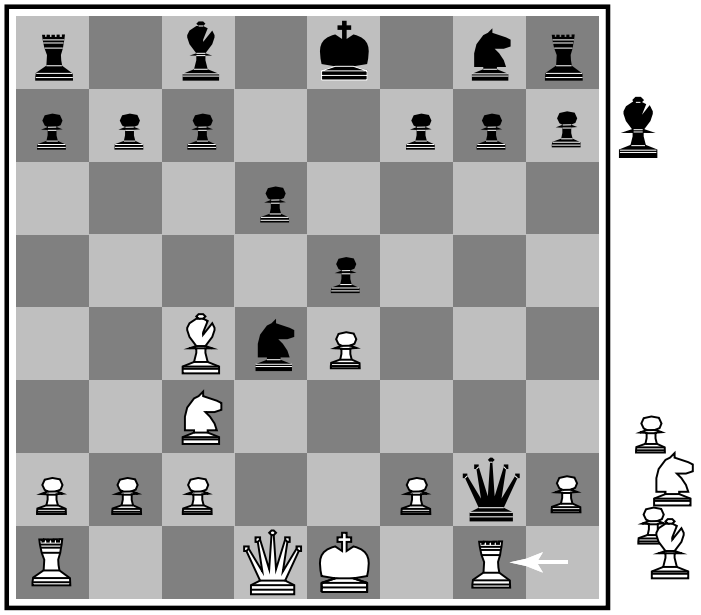
12-warstwowe przeszukiwanie $\approx$ Deep Blue, Kasparov

Przeszukiwanie stabilne

Problem: Stany mają taką samą wartość oceny (na korzyść czarnych),
ale stan z prawej *niestabilny* :
kolejny ruch daje dużą zmianę oceny stanu gry (na korzyść białych)



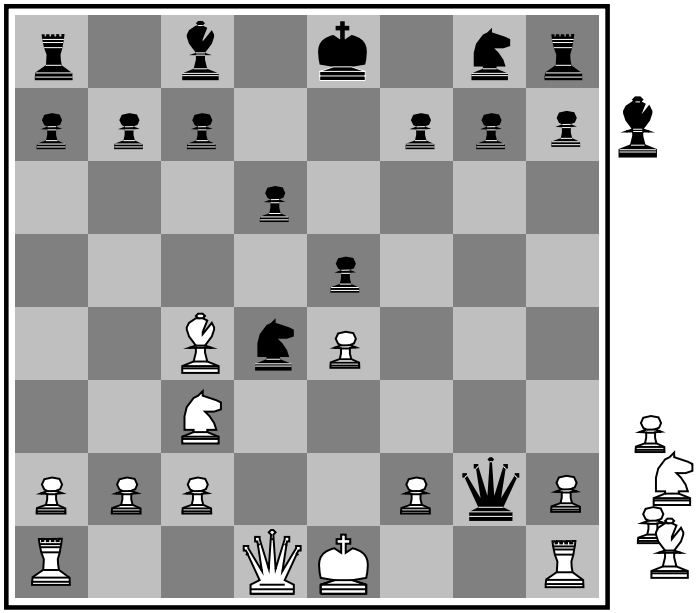
(a) White to move



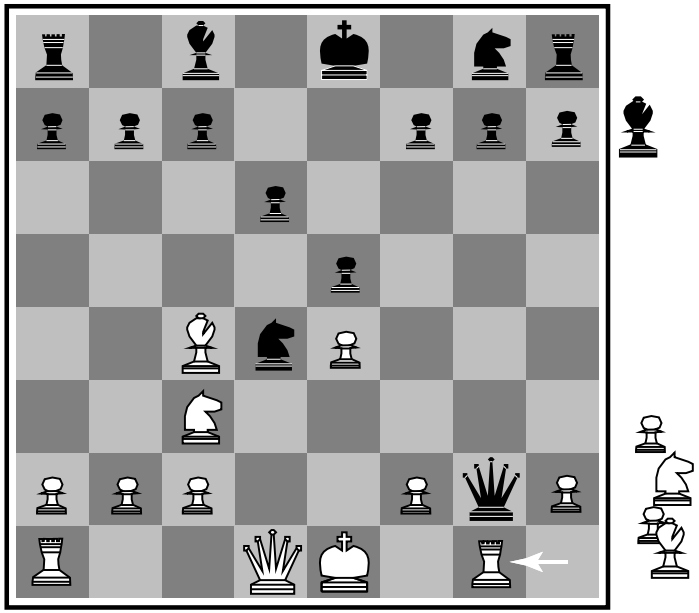
(b) White to move

Przeszukiwanie stabilne

Problem: Stany mają taką samą wartość oceny (na korzyść czarnych),
ale stan z prawej *niestabilny* :
kolejny ruch daje dużą zmianę oceny stanu gry (na korzyść białych)



(a) White to move



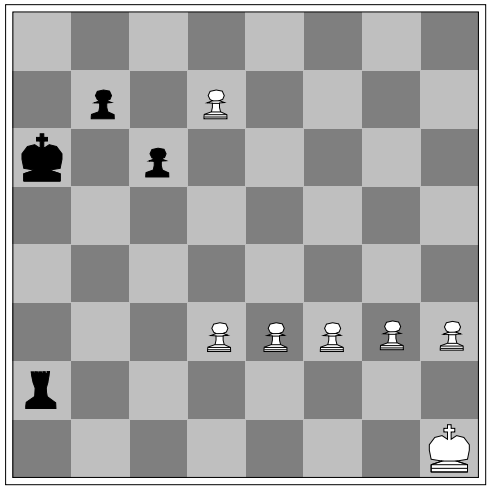
(b) White to move

Rozwiązanie: przeszukiwanie *stabilne*
stany niestabilne są rozwijane do momentu osiągnięcia stanu stabilnego

Przeszukiwanie z pojedynczym rozwinieciem

Efekt horyzontu: gracz wykonuje ruchy odsuwając nieuniknione posunięcie na korzyść przeciwnika poza *horyzont* przeszukiwania

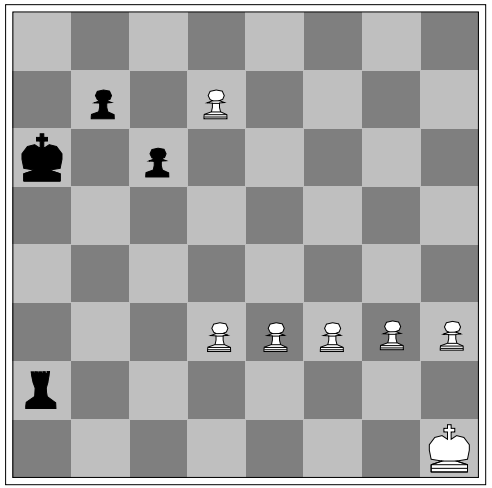
np. czarna wieża powtarza szachowanie białego króla



Black to move

Przeszukiwanie z pojedynczym rozwinieciem

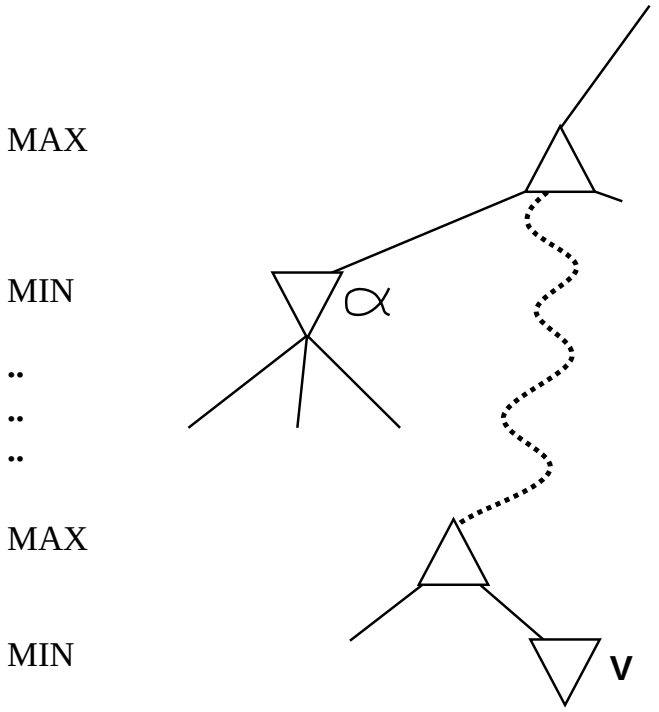
Efekt horyzontu: gracz wykonuje ruchy odsuwając nieuniknione posunięcie
na korzyść przeciwnika poza *horyzont* przeszukiwania
np. czarna wieża powtarza szachowanie białego króla



Black to move

Rozwiązanie: przeszukiwanie *z pojedynczym rozwinięciem*
algorytm wykonuje pogłębione przeszukiwanie
dla wybranych posunięć "wyraźnie lepszych" od pozostałych

Strategia minimax z odcieciem α–β



α jest najlepszą wartością dla MAX poza bieżącą ścieżką przeszukiwania

Jeśli V jest gorsze niż α, MAX nigdy nie wejdzie do tej gałęzi
$\Rightarrow$ gałąź z V można odciąć

β jest definiowane analogicznie dla MIN

Strategia minimax z odcieciem α–β: algorytm

function ALPHA-BETA-SEARCH($state, game$) **returns** an action
  $action$, $state \leftarrow$ the a, s **in** SUCCESSORS[$game$]($state$)
    such that MIN-VALUE($s, game, -\infty, +\infty$) is maximized
  return $action$

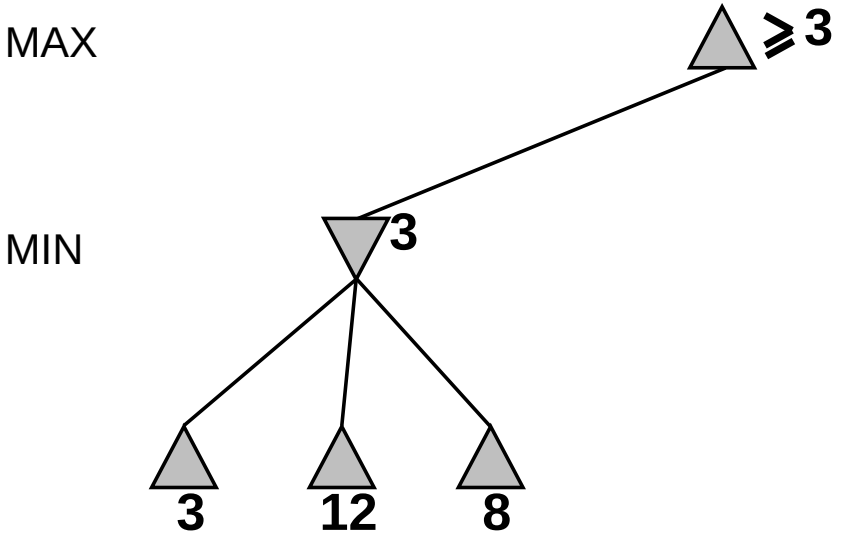
---

function MAX-VALUE($state, game, \alpha, \beta$) **returns** the minimax value of $state$
  if CUTOFF-TEST($state$) **then return** EVAL($state$)
  for each s **in** SUCCESSORS($state$) **do**
    $\alpha \leftarrow \max(\alpha,$ MIN-VALUE($s, game, \alpha, \beta$))
    if $\alpha \geq \beta$ **then return** β
  return α

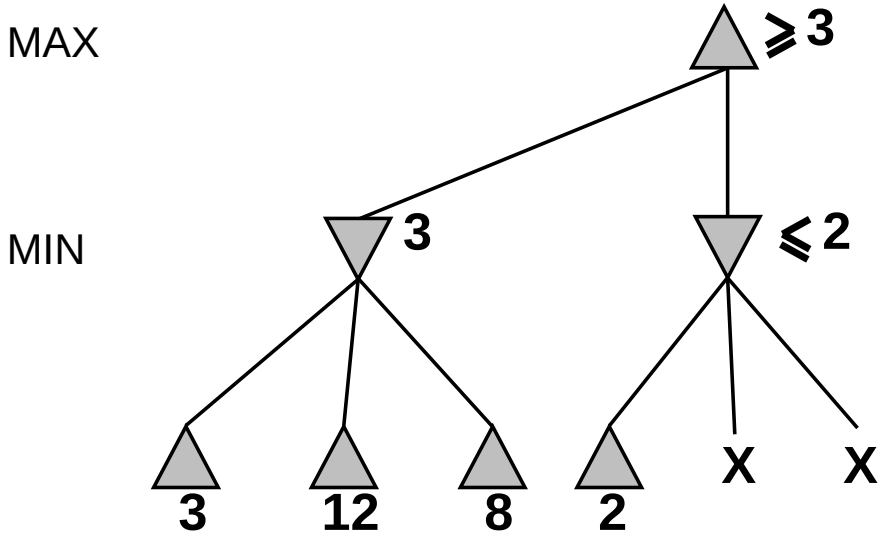
---

function MIN-VALUE($state, game, \alpha, \beta$) **returns** the minimax value of $state$
  if CUTOFF-TEST($state$) **then return** EVAL($state$)
  for each s **in** SUCCESSORS($state$) **do**
    $\beta \leftarrow \min(\beta,$ MAX-VALUE($s, game, \alpha, \beta$))
    if $\beta \leq \alpha$ **then return** α
  return β

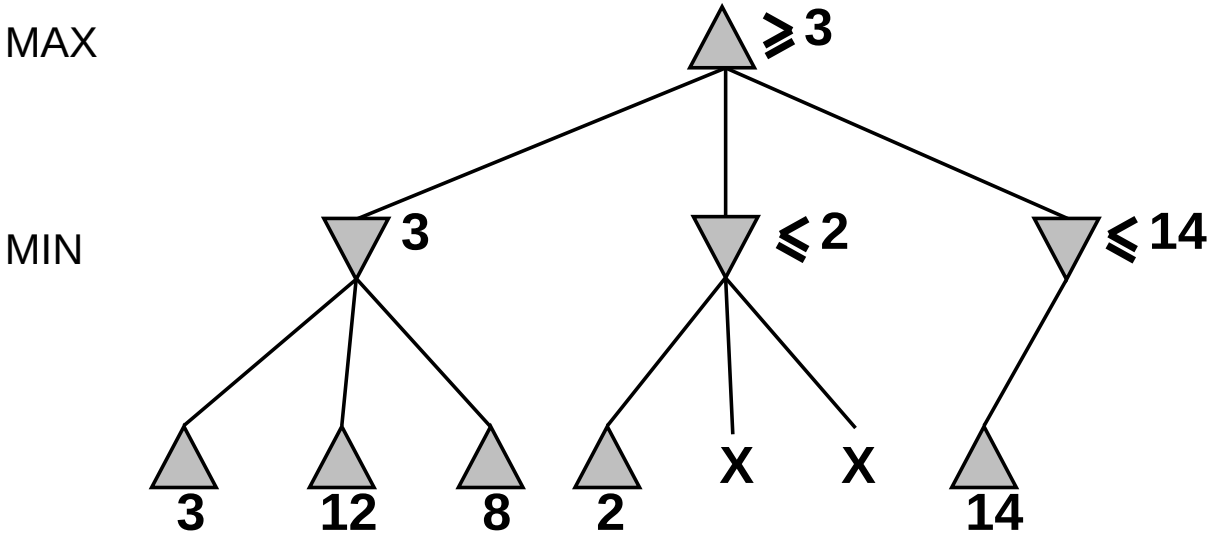
Strategia minimax z odcieciem α–β: przyklad



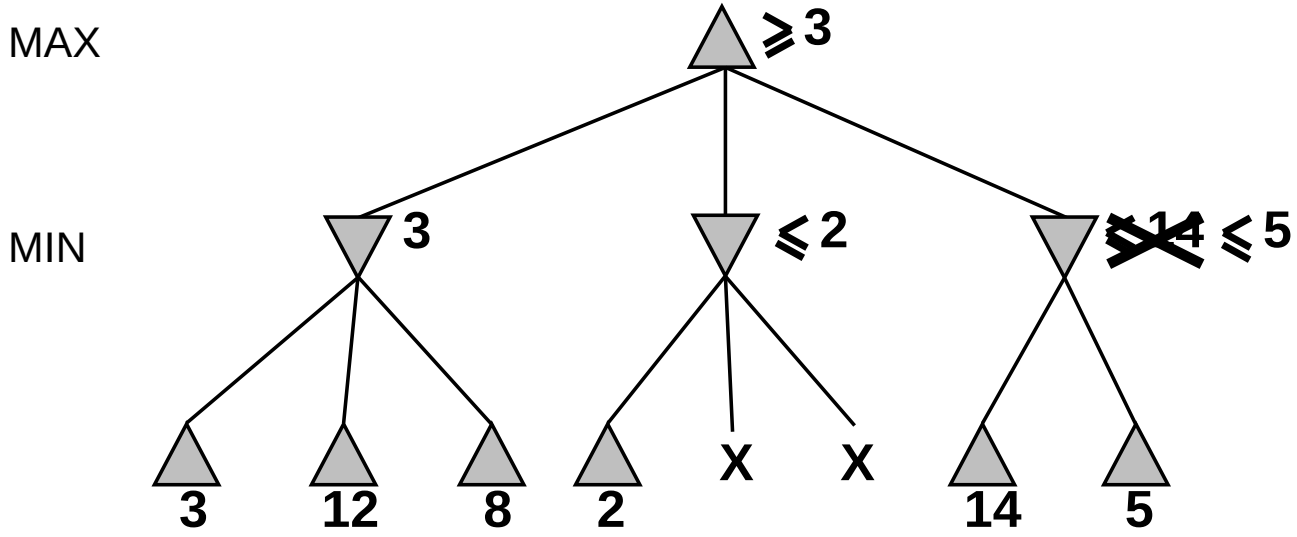
Strategia minimax z odcieciem α–β: przyklad



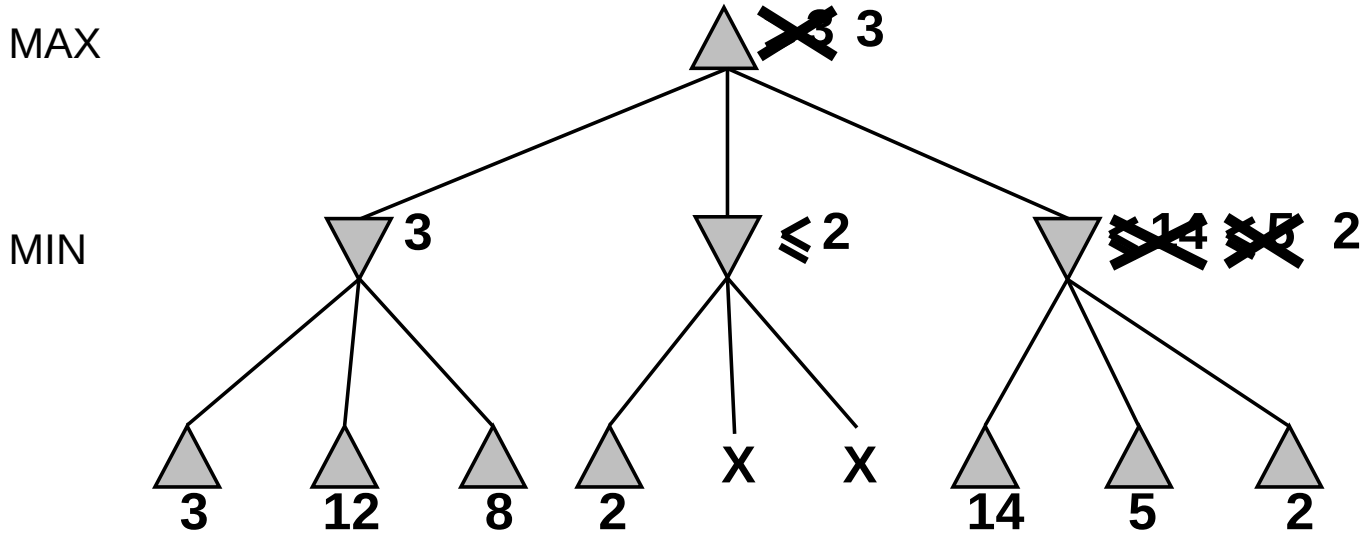
Strategia minimax z odcieciem α–β: przyklad



Strategia minimax z odcieciem α–β: przyklad



Strategia minimax z odcieciem α–β: przyklad



Strategia minimax z odcieciem α–β: wlasnosci

Odcinanie jest "czyste": nie ma wpływu na optymalność i wynik przeszukiwania

Właściwe uporządkowanie posunięć poprawia efektywność odcinania

Dla "perfekcyjnego uporządkowania" posunięć złożoność czasowa $= O(b^{m/2})$

$\Rightarrow$ *podwaja* głębokość przeszukiwania

$\Rightarrow$ może łatwo zejść do 8-ego poziomu i grać dobre szachy

Gry deterministyczne: osiagniecia

Warcaby: Chinook zakończył 40-letnie panowanie mistrza świata Mariona Tinsley w 1994. Użył biblioteki wszystkich zakończeń dla 8 lub mniej pionków na planszy, w sumie 443,748,401,247 pozycji.

Szachy: Deep Blue pokonał mistrza świata Gary Kasparowa w meczu z 6-ioma partiami w 1997. Deep Blue przeszukiwał 200 milionów pozycji na sekundę, używając bardzo wyszukanej funkcji oceny, i nieznanych metod rozszerzających niektóre ścieżki przeszukiwania do głębokości 40.

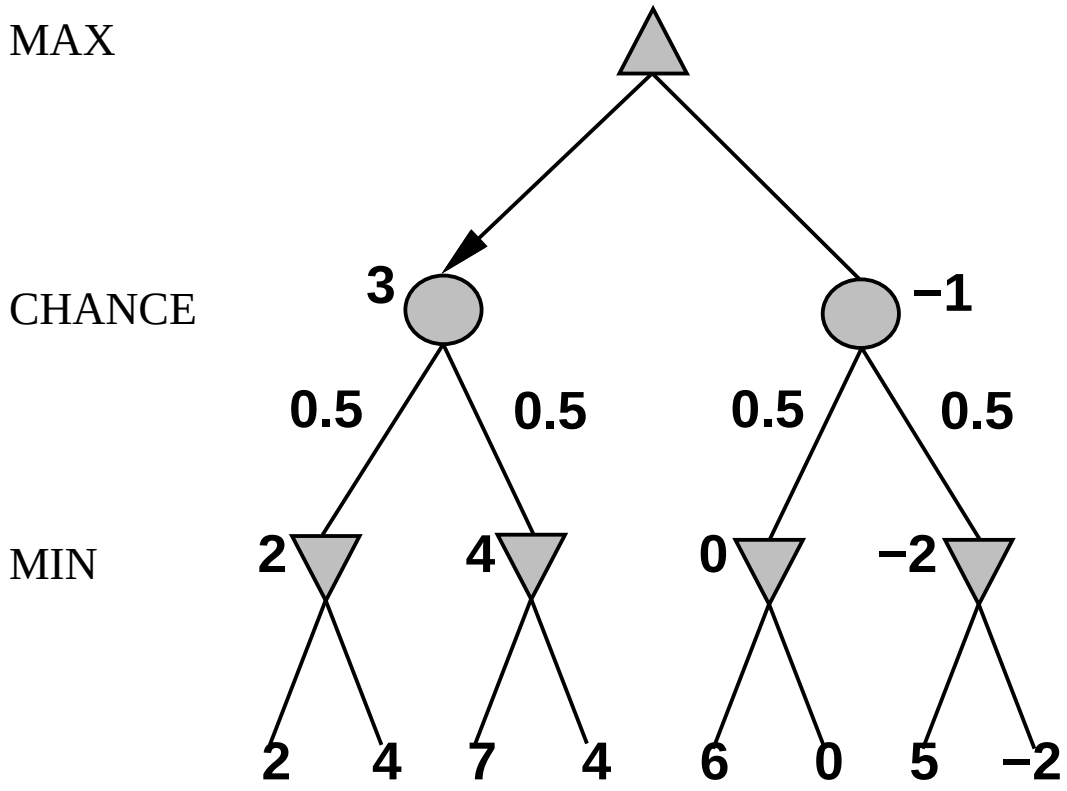
Otello: mistrz świata odmówił rozgrywki z komputerami, które są zbyt silne.

Go: mistrz świata odmówił rozgrywki z komputerami, które są zbyt słabe. W go, $b > 300$, więc większość programów używa bazy wiedzy z wzorcami do wyboru dopuszczalnych ruchów.

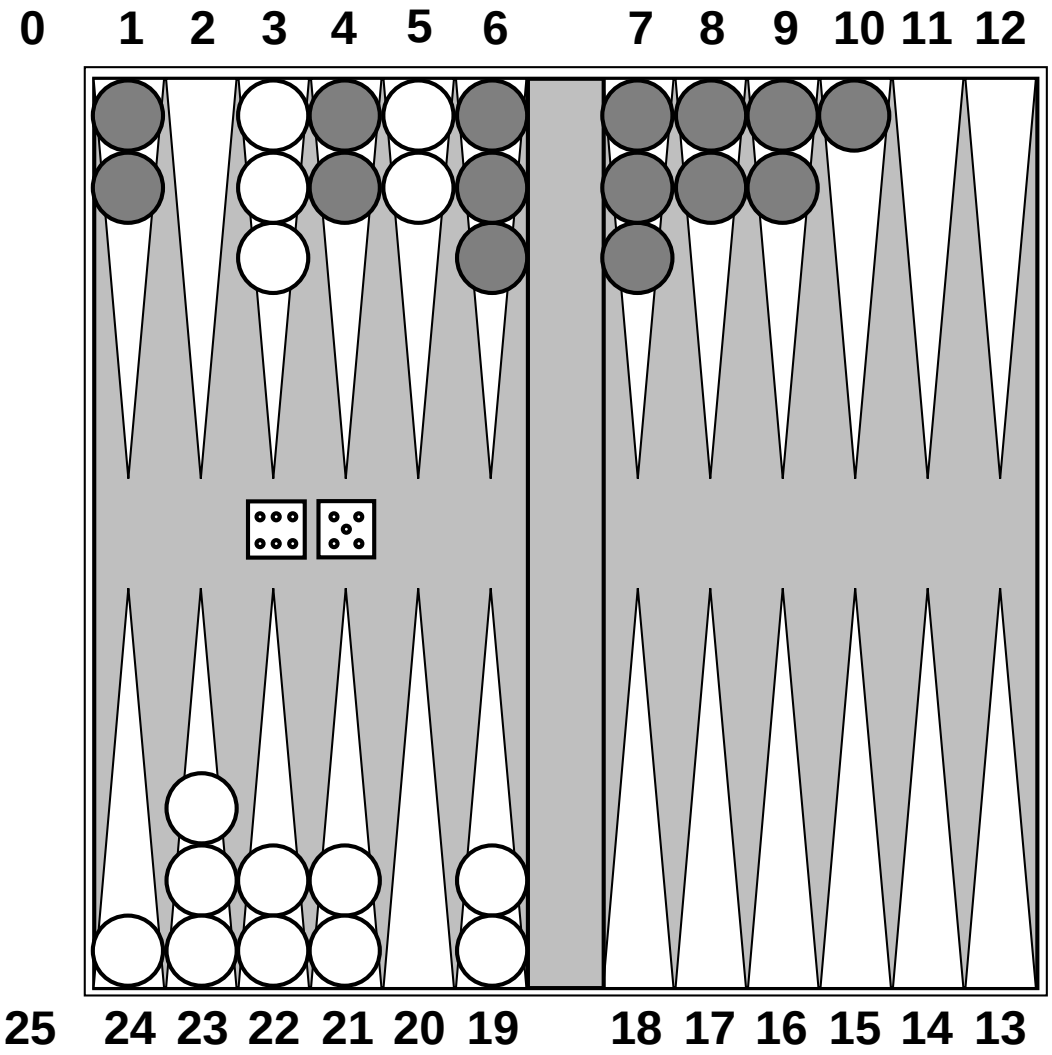
Gry niedeterministyczne

Źródło niedeterminizmu: rzut kostką, tasowanie kart

Przykład z rzucaniem monetą:



Gry niedeterministyczne: backgammon



Strategia usrednionego minimax

Uogólnienie strategii minimax dla gier niedeterministycznych

function EXPECTIMINIMAX-DECISION(*state*, *game*) **returns** *an action*

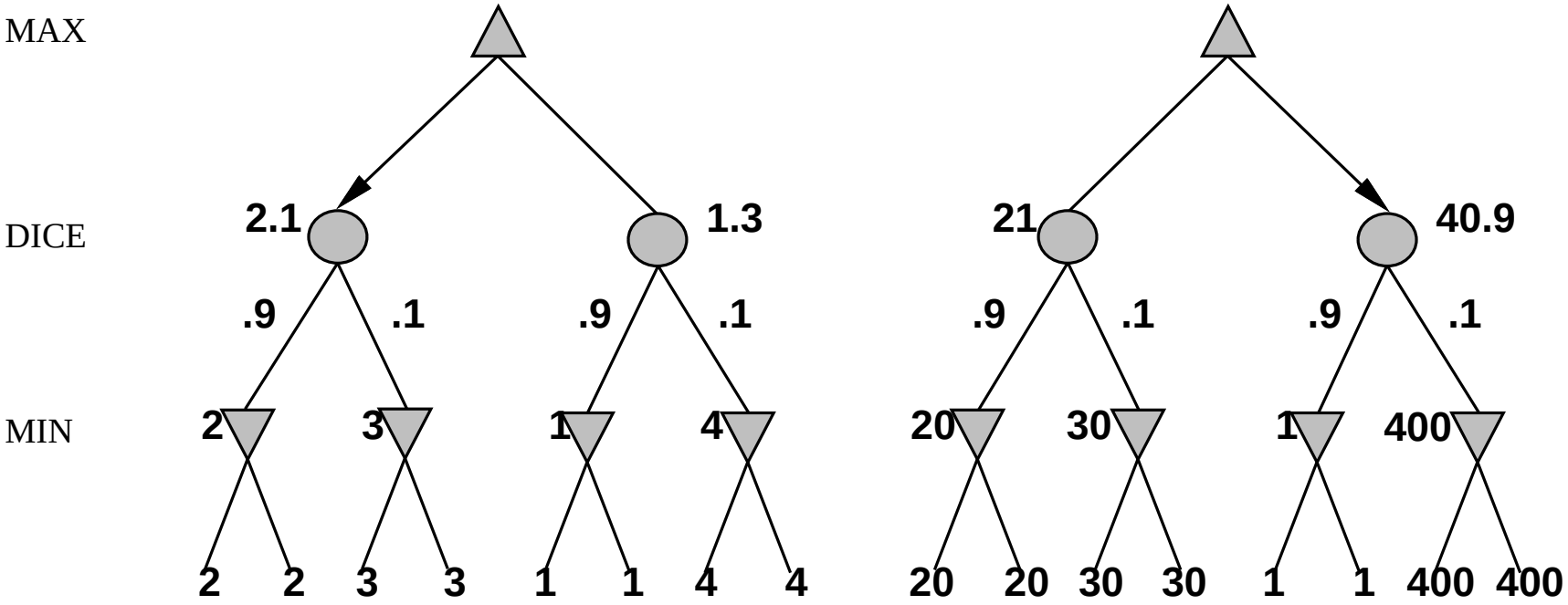
action, *state* ← the *a*, *s* **in** SUCCESSORS(*state*)
such that EXPECTIMINIMAX-VALUE(*s*, *game*) is maximized
return *action*

---

function EXPECTIMINIMAX-VALUE(*state*, *game*) **returns** *a utility value*

if TERMINAL-TEST(*state*) **then**
return UTILITY(*state*)
else if *state* is a MAX node **then**
return the highest EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)
else if *state* is a MIN node **then**
return the lowest EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)
else if *state* is a chance node **then**
return average of EXPECTIMINIMAX-VALUE of SUCCESSORS(*state*)

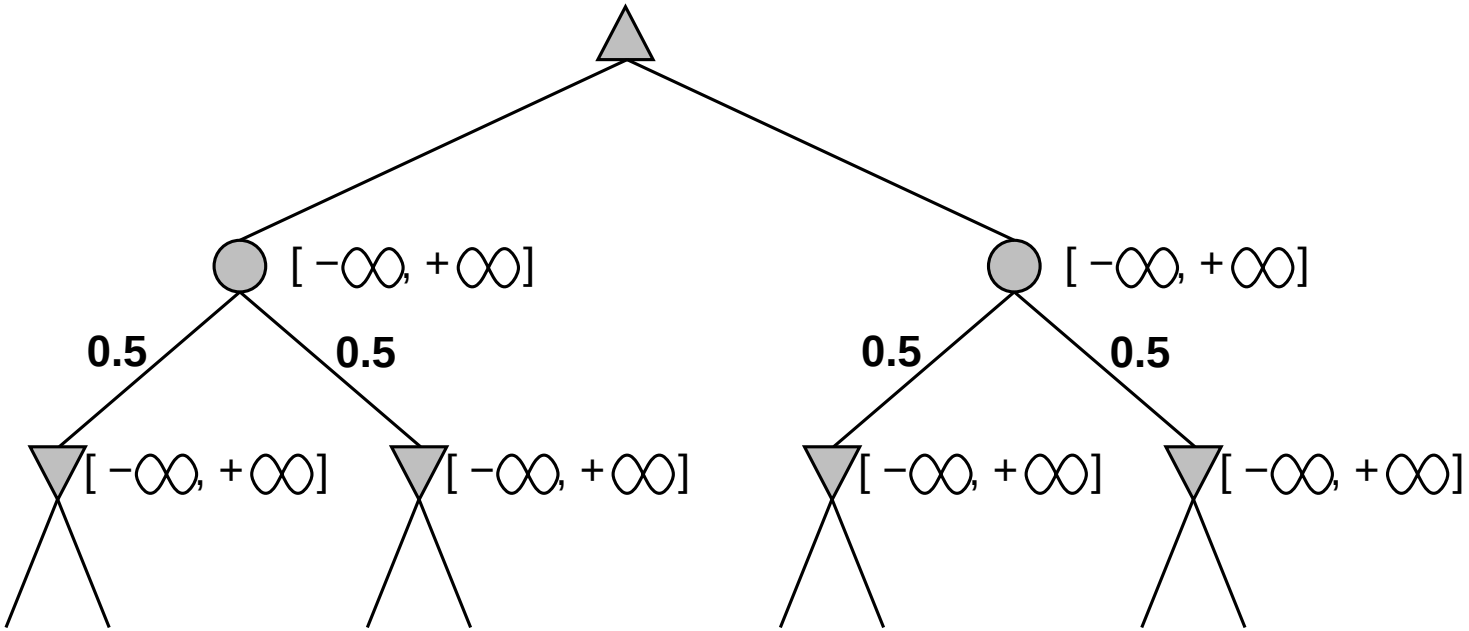
Strategia usrednionego minimax: wlasnosci



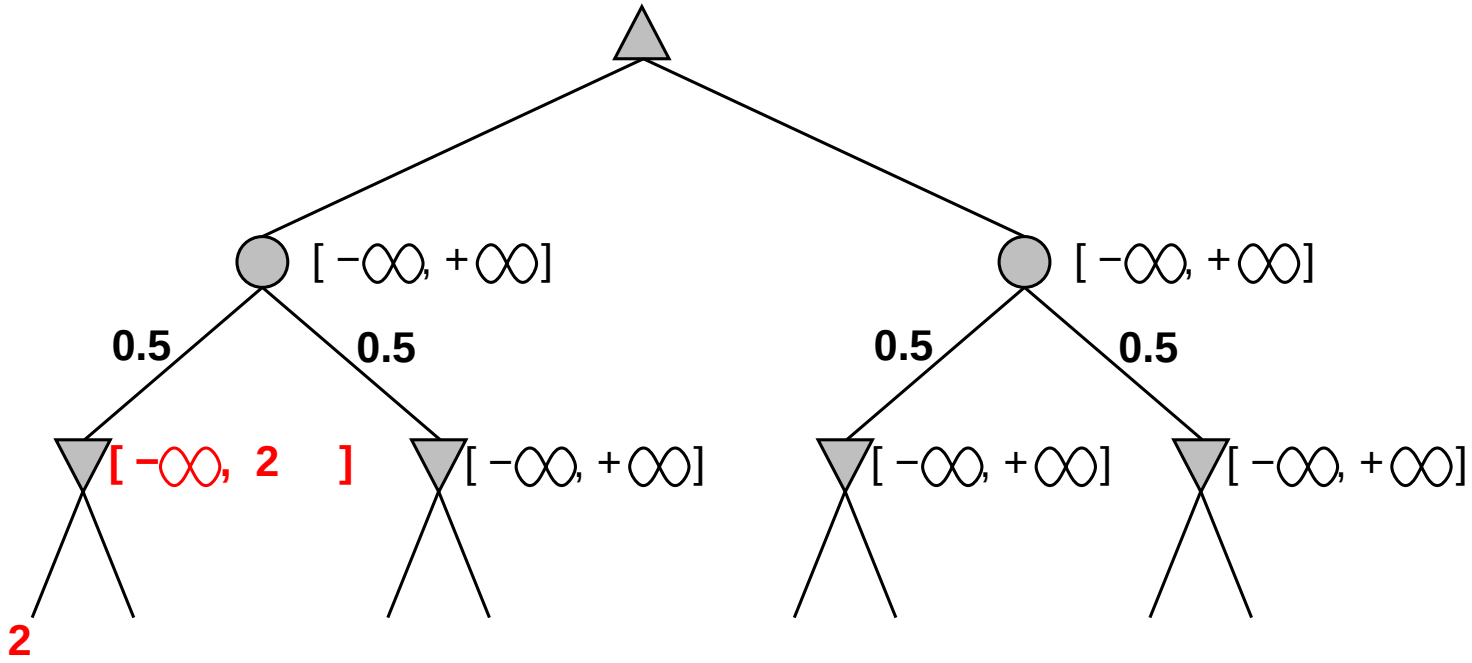
Działanie funkcji oceny EVAL jest zachowane tylko dla *dodatnich liniowych* przekształceń tej funkcji

Stąd EVAL powinna być propocjonalna do wartości oczekiwanej wypłaty

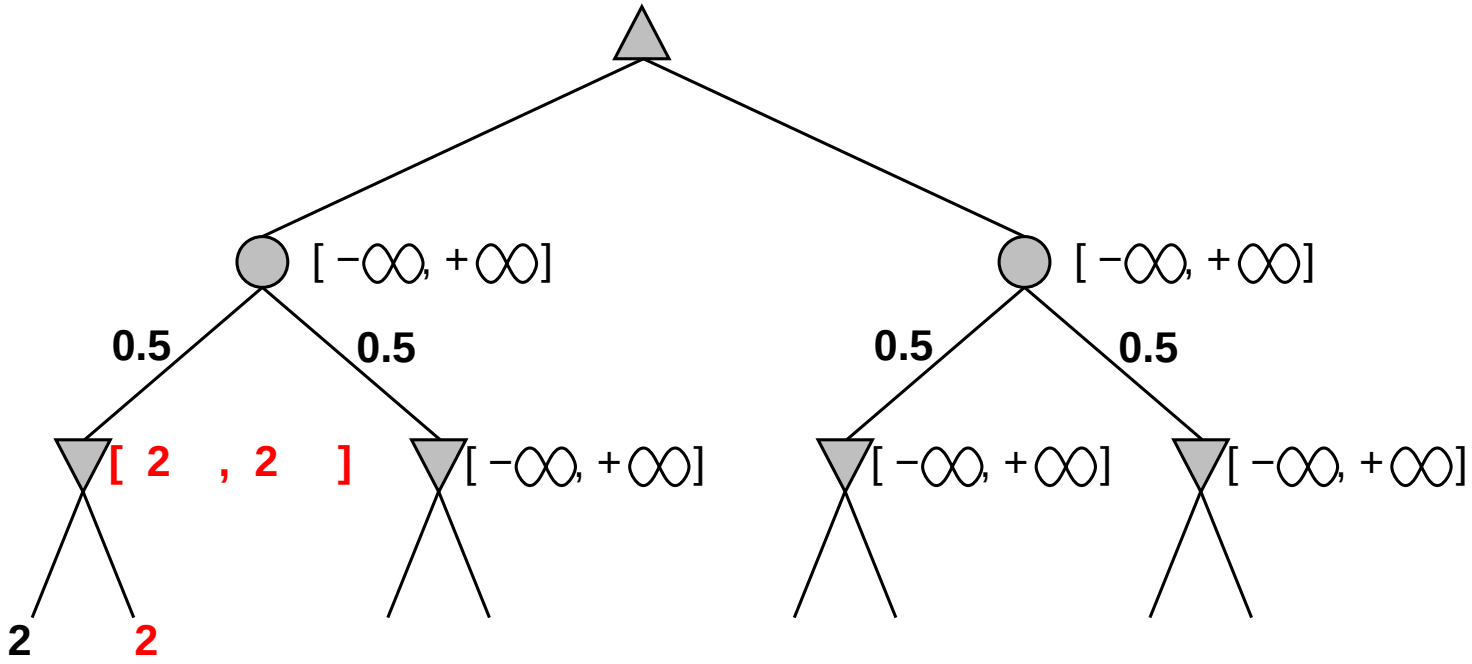
Strategia usrednionego minimax z odcieciem α-β



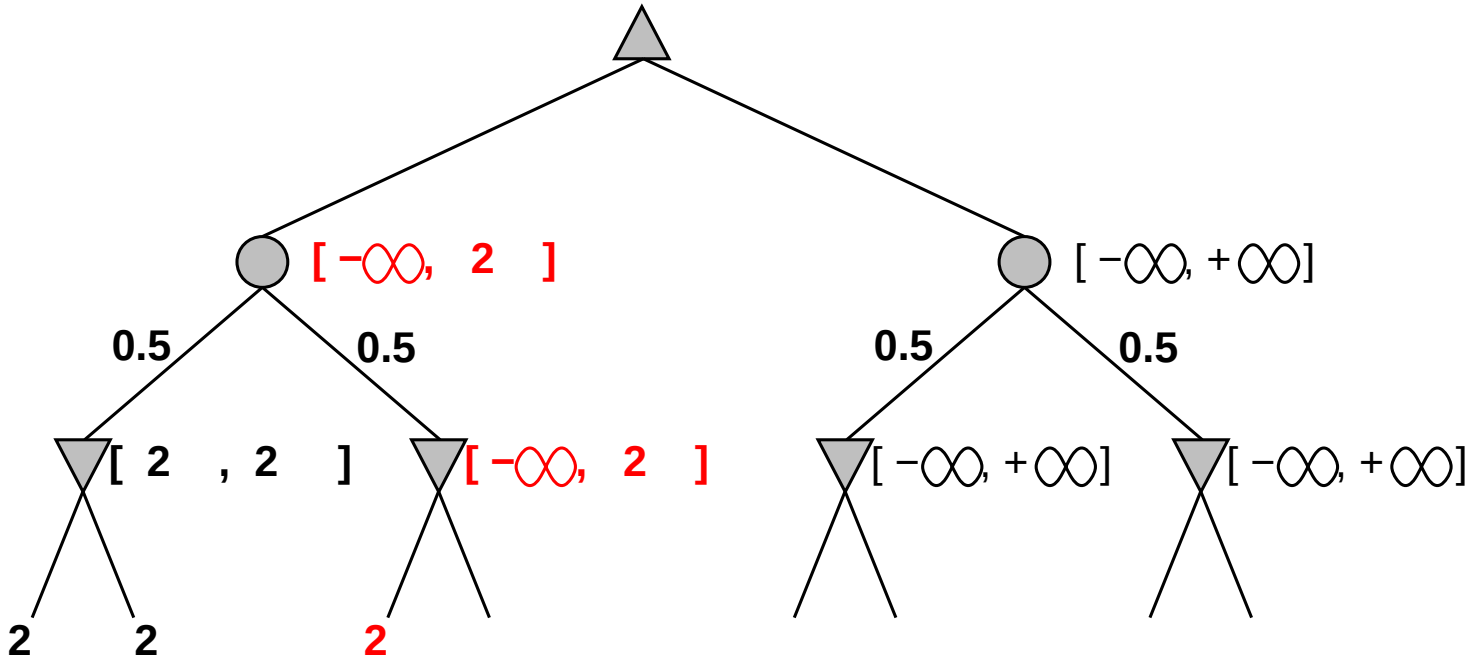
Strategia usrednionego minimax z odcieciem α-β



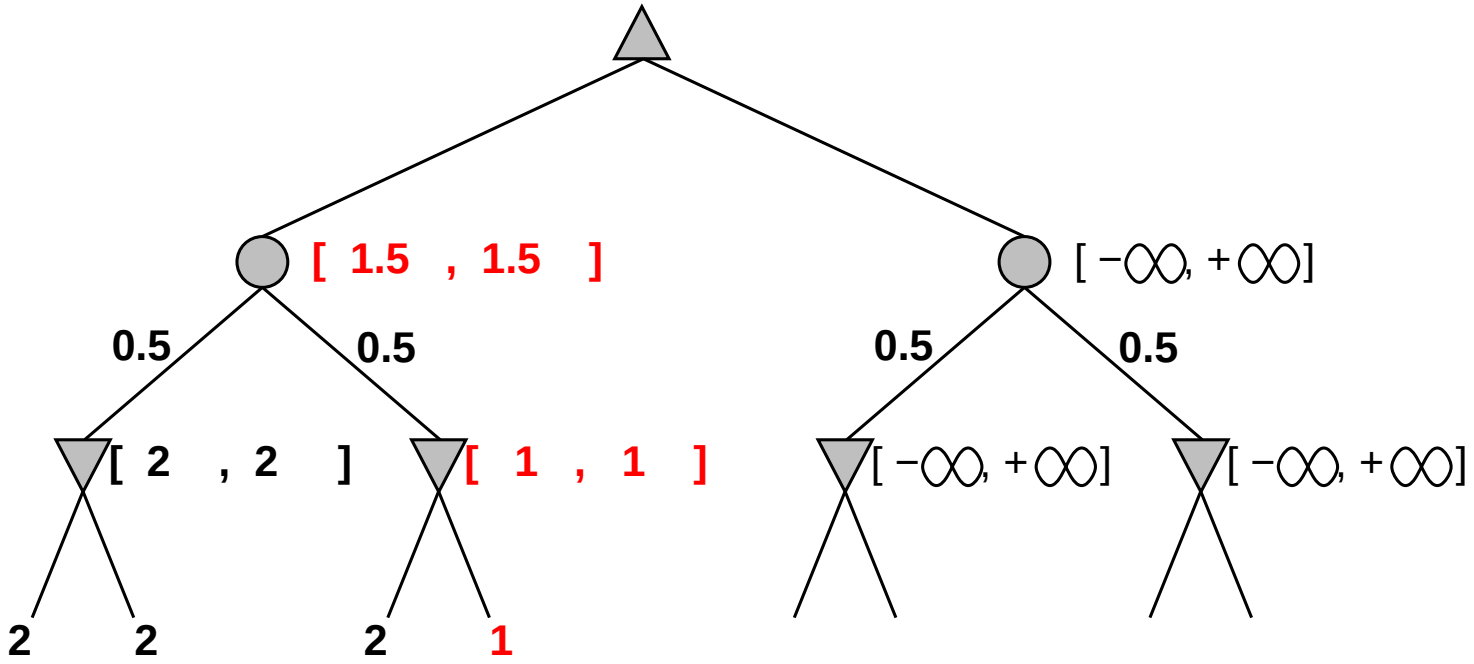
Strategia usrednionego minimax z odcieciem α-β



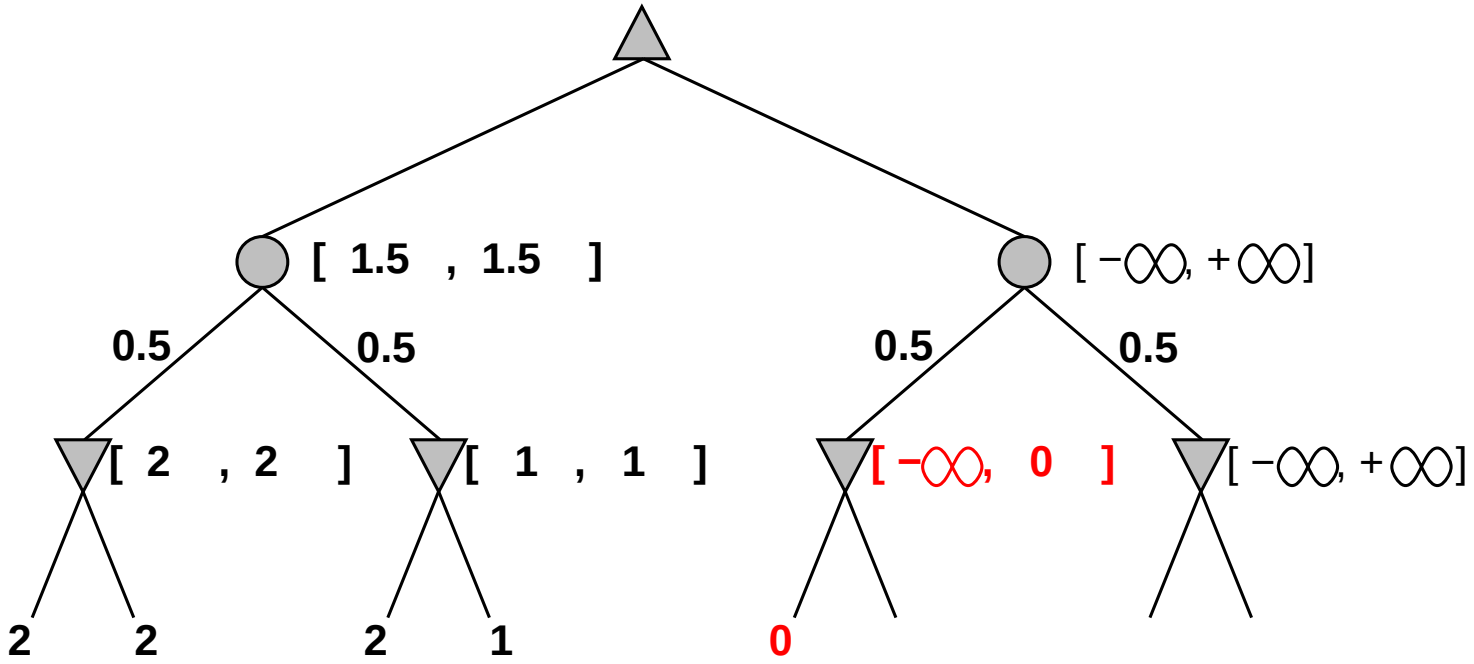
Strategia usrednionego minimax z odcieciem α-β



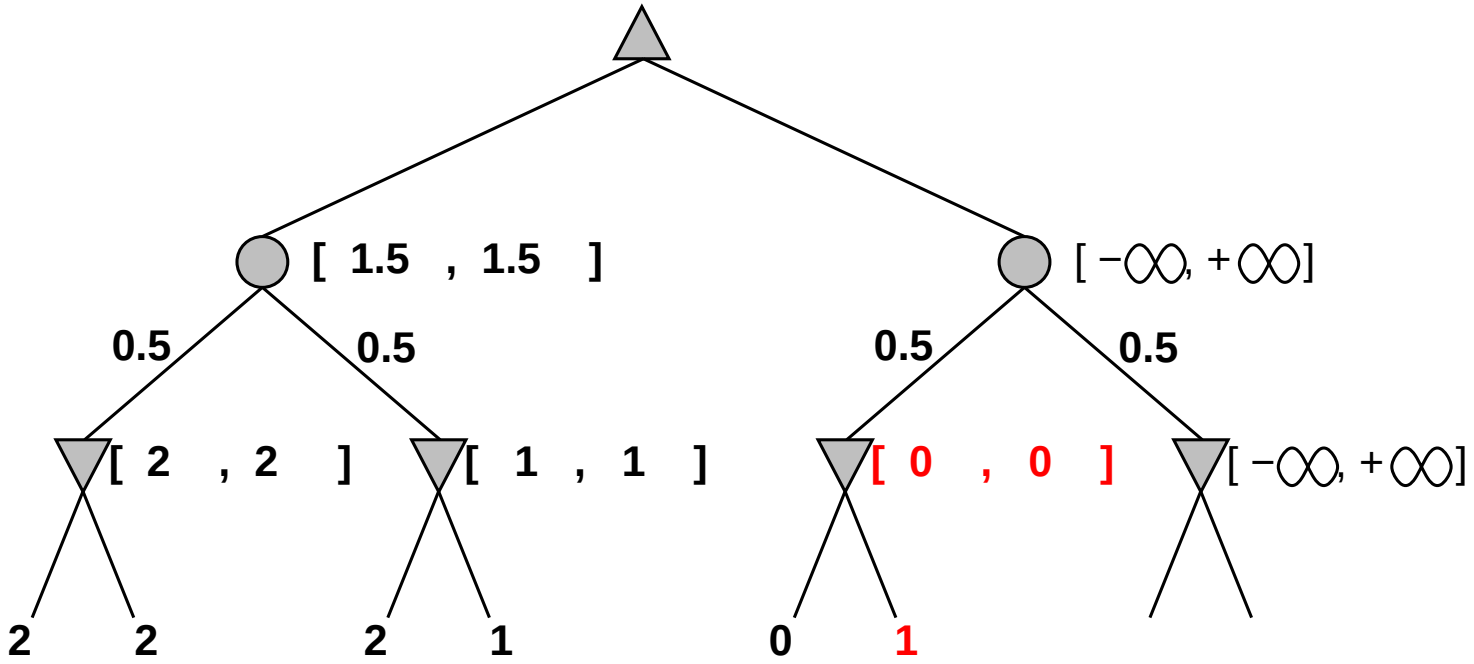
Strategia usrednionego minimax z odcieciem α-β



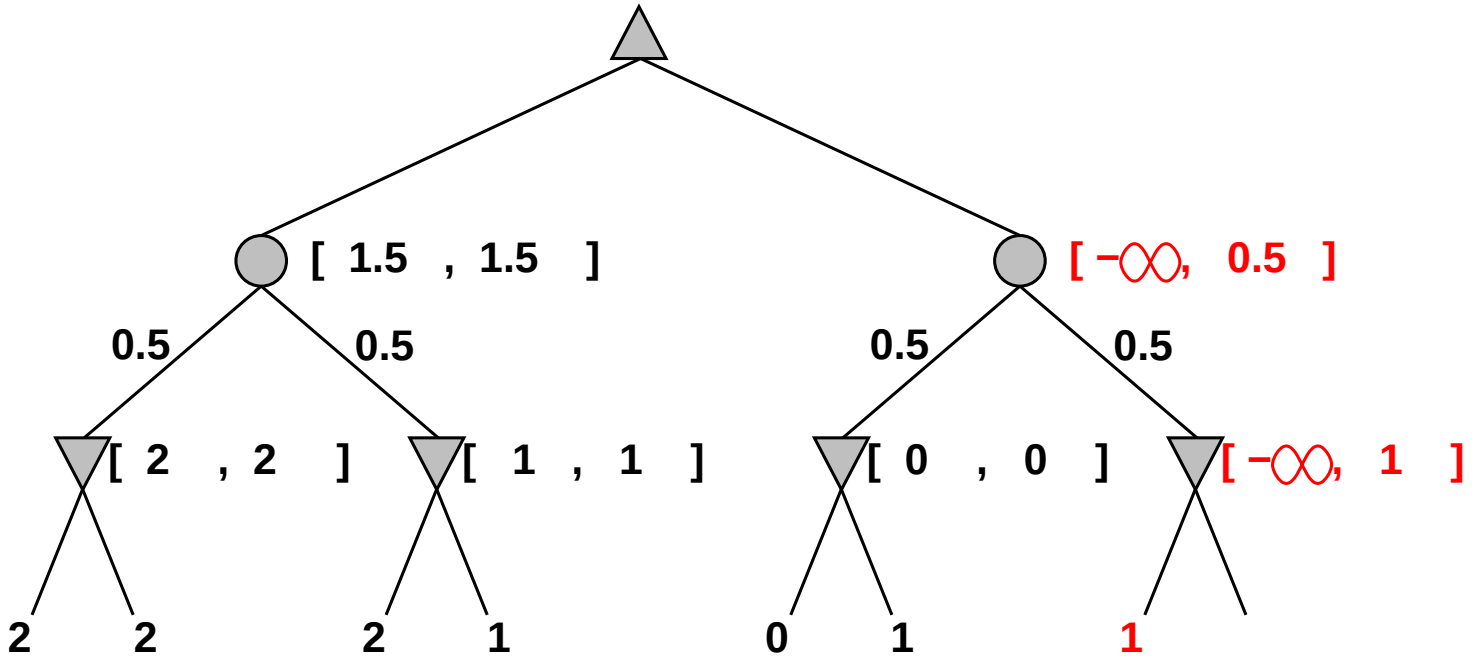
Strategia usrednionego minimax z odcieciem α-β



Strategia usrednionego minimax z odcieciem α-β

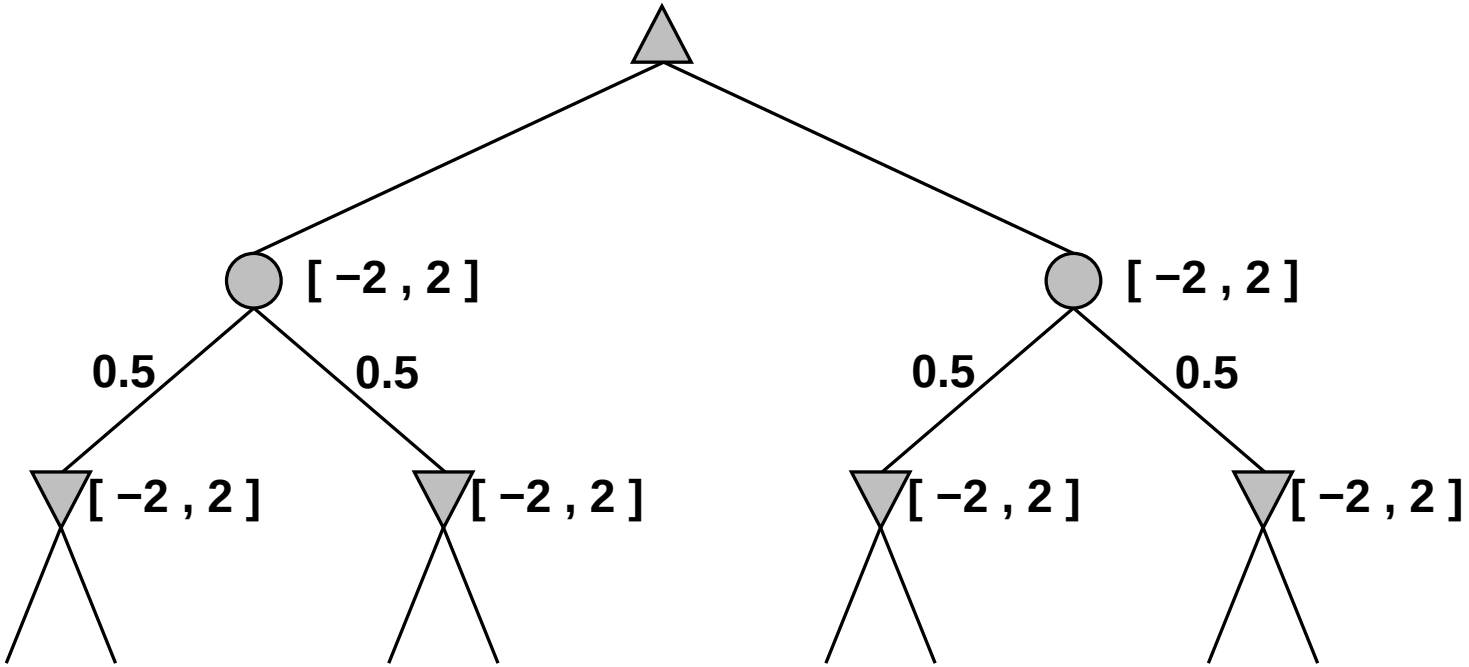


Strategia usrednionego minimax z odcieciem α-β



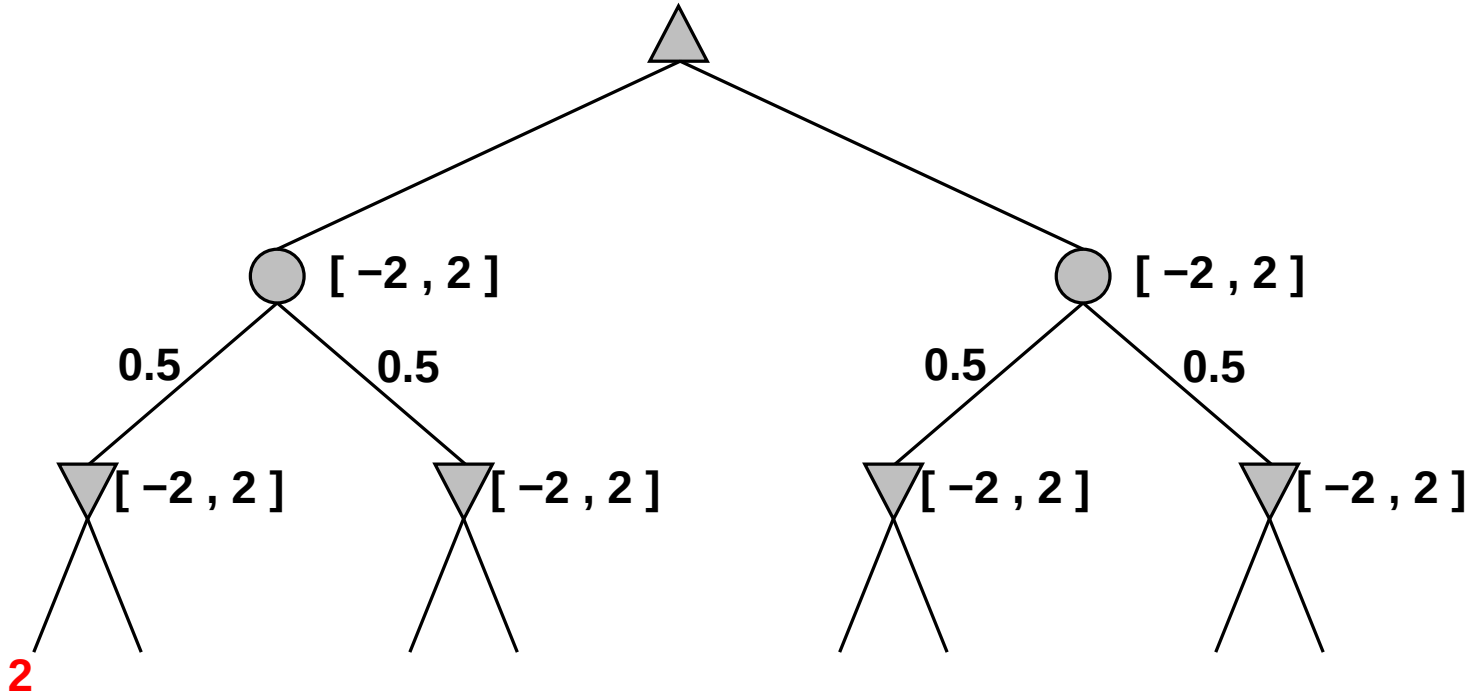
Strategia usrednionego minimax z odcieciem α-β

Bardziej efektywna, jeśli wartość wypłaty ograniczona



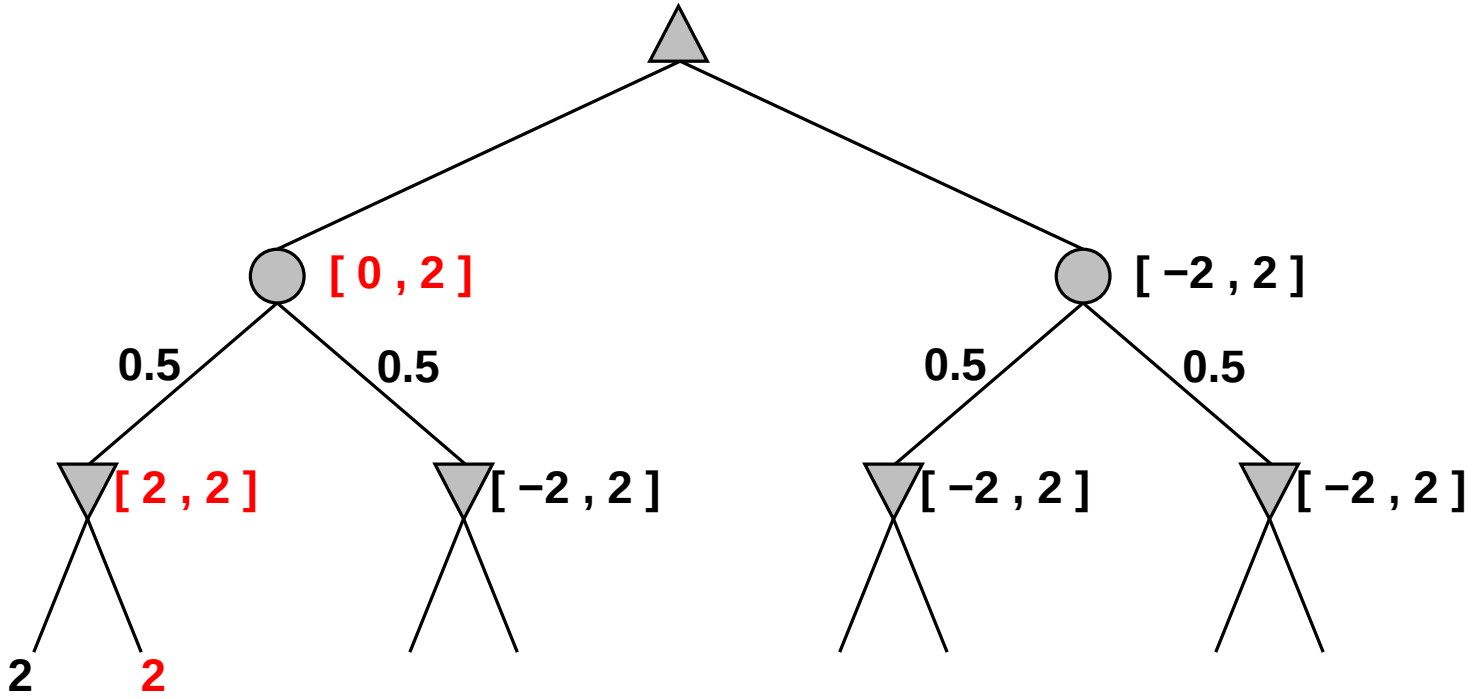
Strategia usrednionego minimax z odcieciem α-β

Bardziej efektywna, jeśli wartość wypłaty ograniczona



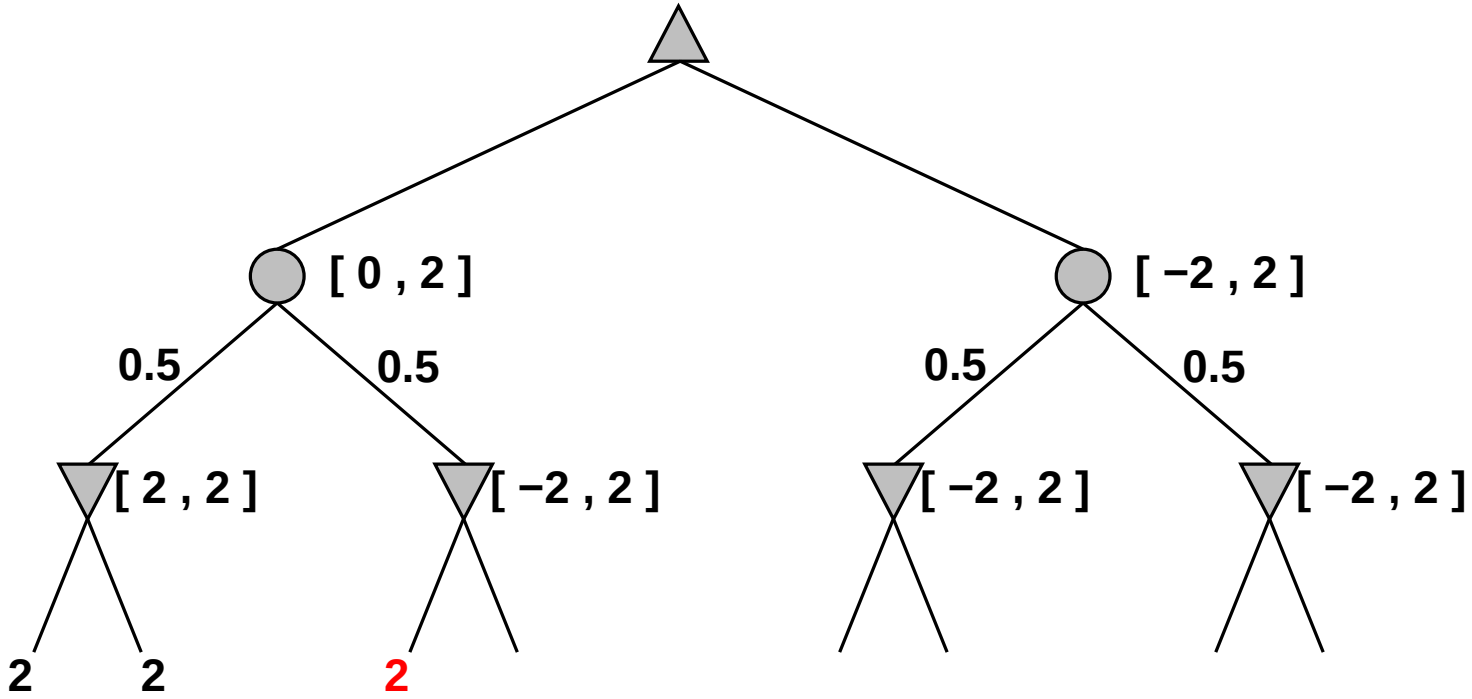
Strategia usrednionego minimax z odcieciem α-β

Bardziej efektywna, jeśli wartość wypłaty ograniczona



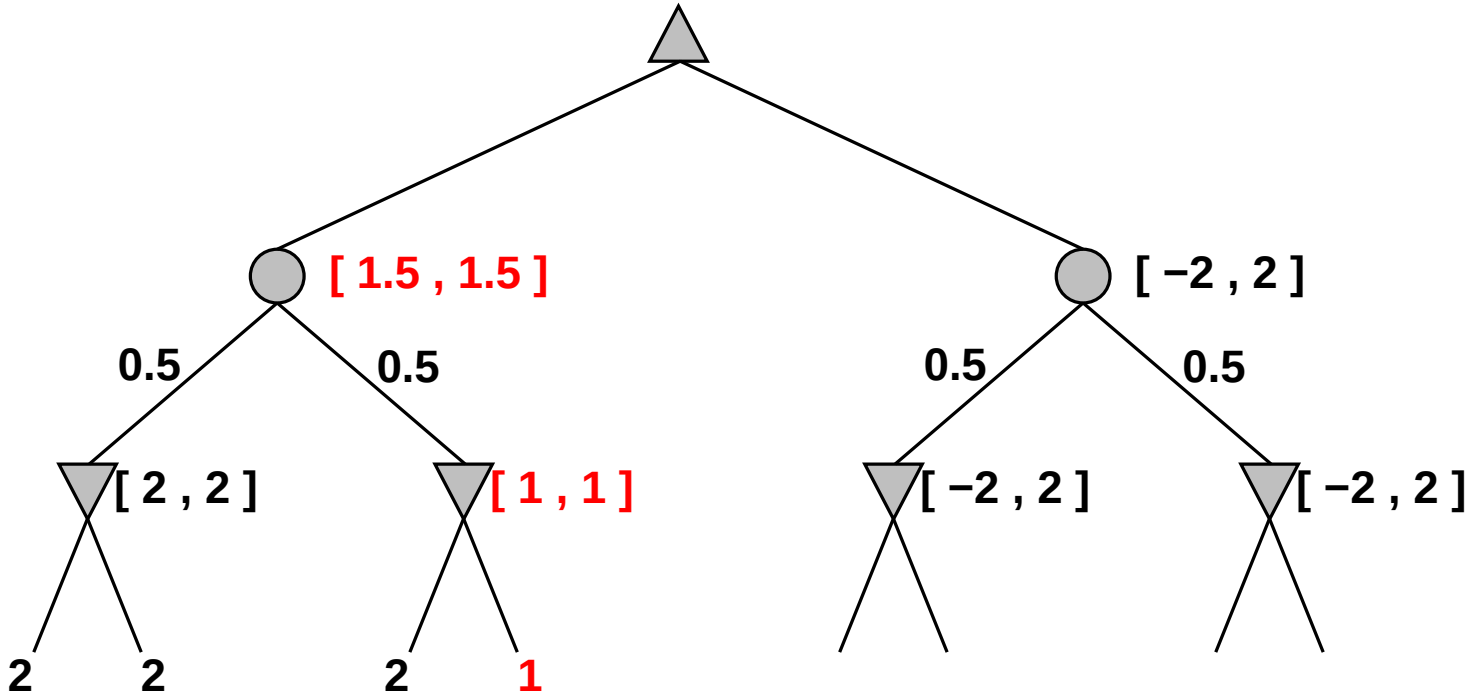
Strategia usrednionego minimax z odcieciem α-β

Bardziej efektywna, jeśli wartość wypłaty ograniczona



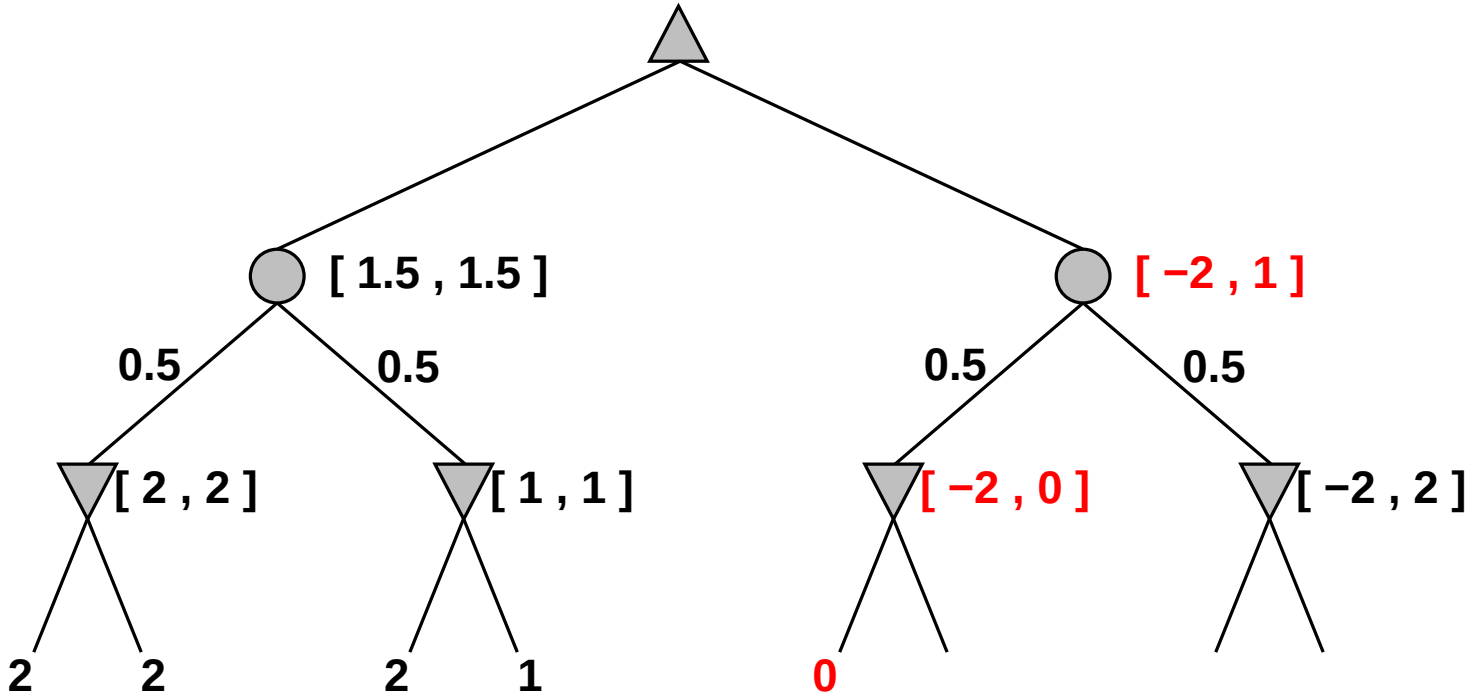
Strategia usrednionego minimax z odcieciem α-β

Bardziej efektywna, jeśli wartość wypłaty ograniczona



Strategia usrednionego minimax z odcieciem α-β

Bardziej efektywna, jeśli wartość wypłaty ograniczona



Gry niedeterministyczne: wlasnosci

Rzuty kostką zwiększają b: 21 możliwych rzutów dla 2 kostek
Backgammon $\approx$ 20 dopuszczalnych posunięć

$$\text{głębokość } 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$$

Jak głębokość wzrasta, prawdopodobieństwo osiągnięcia danego węzła maleje
$\Rightarrow$ wartość sprawdzania wprzód jest nikła

Odcinanie α–β jest dużo mniej efektywne

Program TDGammon:
przeszukiwanie na głębokość 2
+ bardzo dobra funkcja oceny stanu Eval
$\approx$ poziom mistrza świata

Gry z niepelna informacja

Np. gry karciane, w których początkowy zestaw kart przeciwnika jest nieznany

Można policzyć prawdopodobieństwo każdego rozdania
$\Rightarrow$ wygląda jak jeden "duży" rzut kostką na początku gry

Pomysł:

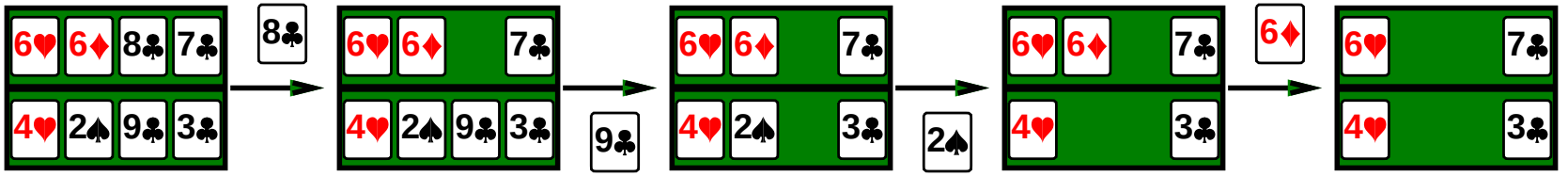
algorytm oblicza wartość minimax dla każdej akcji w każdym możliwym rozdaniu i wybiera akcje z największą wartością uśrednioną po wszystkich rozdaniach

GIB, najlepszy program do brydża, przybliża tą ideę

1) generuje 100 rozdań zgodnych z informacją z licytacji
2) wybiera akcję, która zbiera średnio najwięcej lew

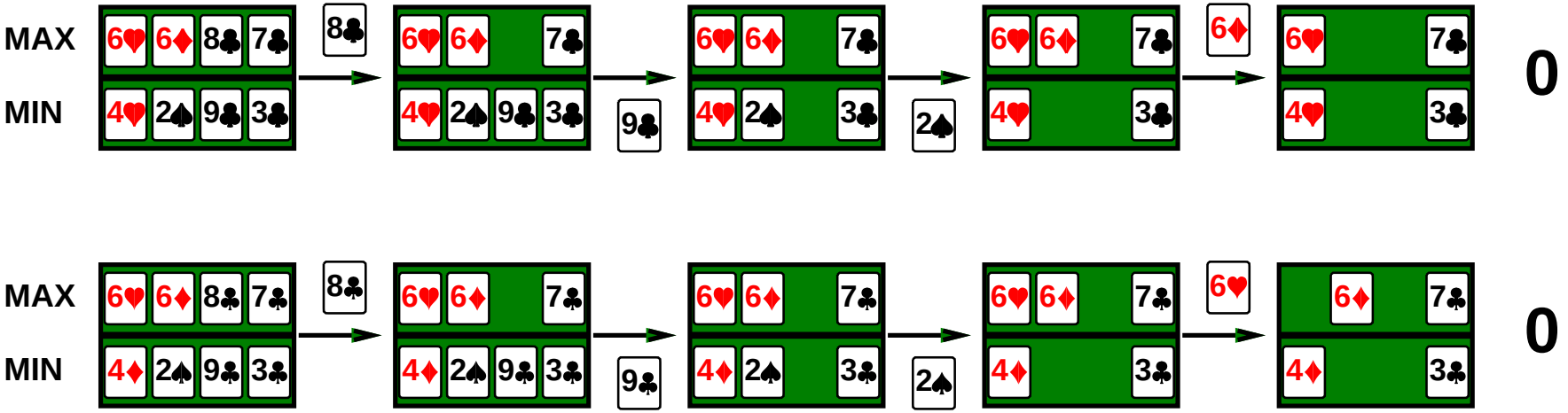
Gry z niepelna informacja: przyklad

Gra w otwarte karty, MAX gra pierwszy



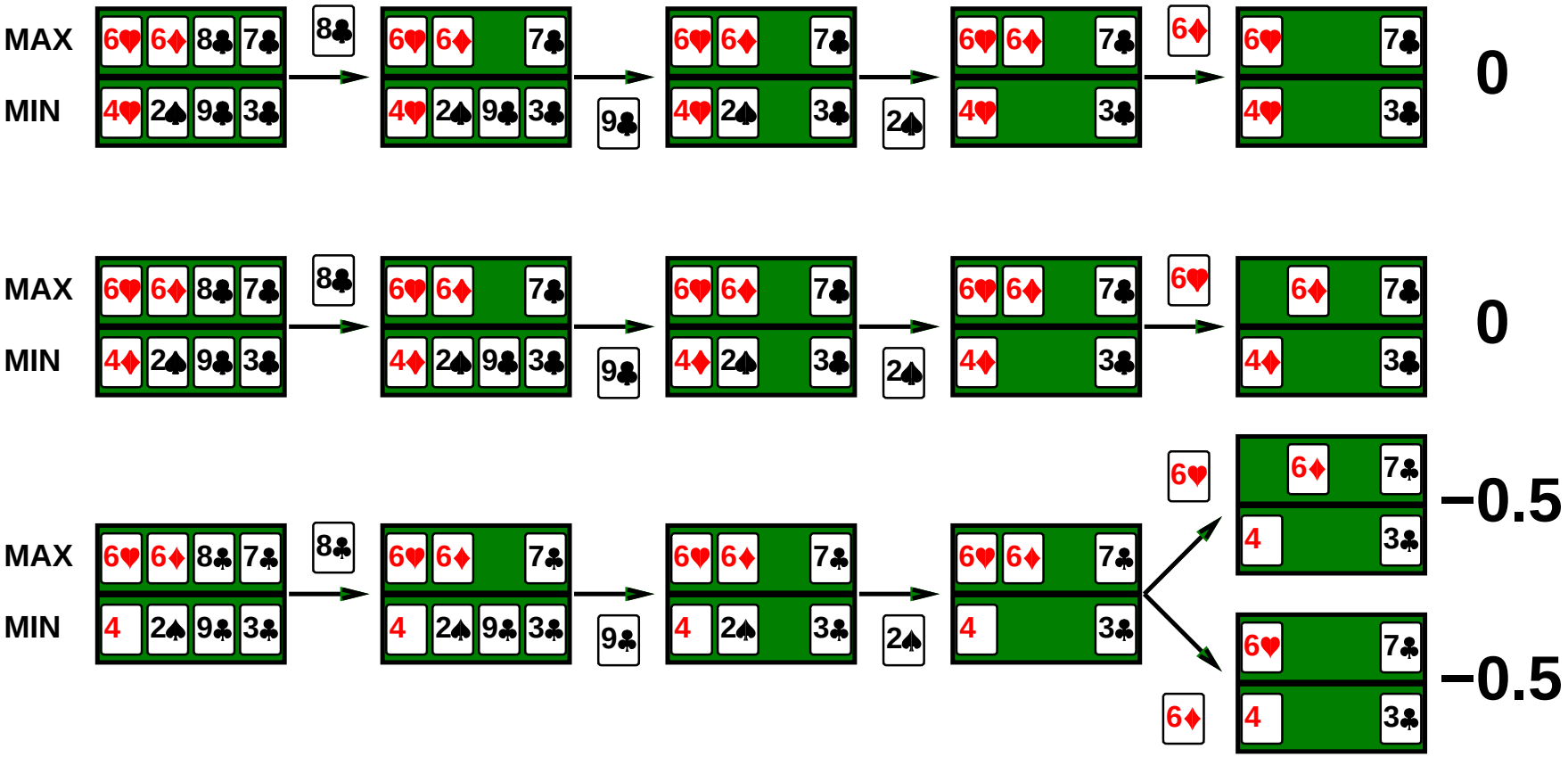
Gry z niepelna informacja: przyklad

Gra w otwarte karty, MAX gra pierwszy



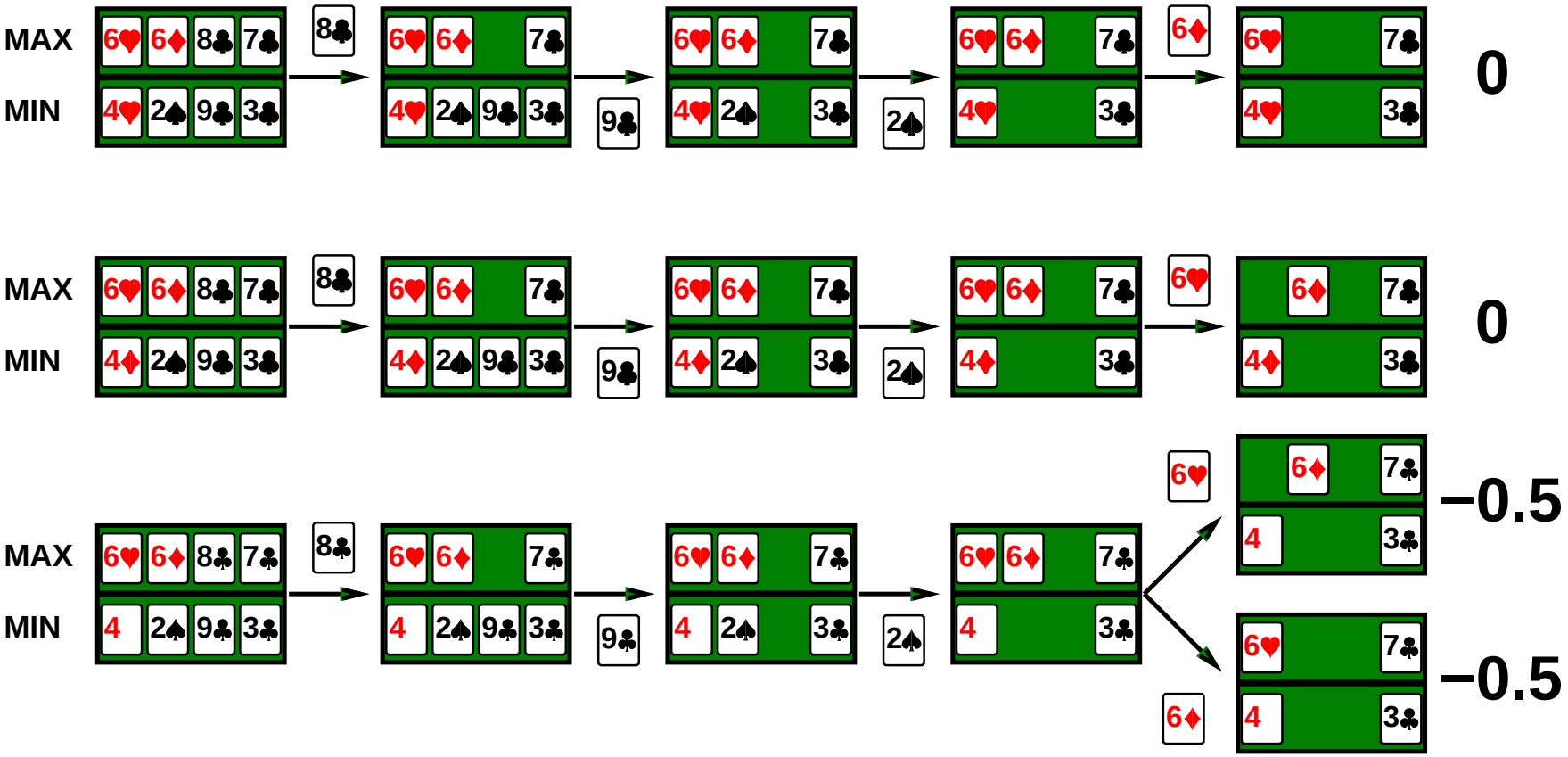
Gry z niepelna informacja: przyklad

Gra w otwarte karty, MAX gra pierwszy



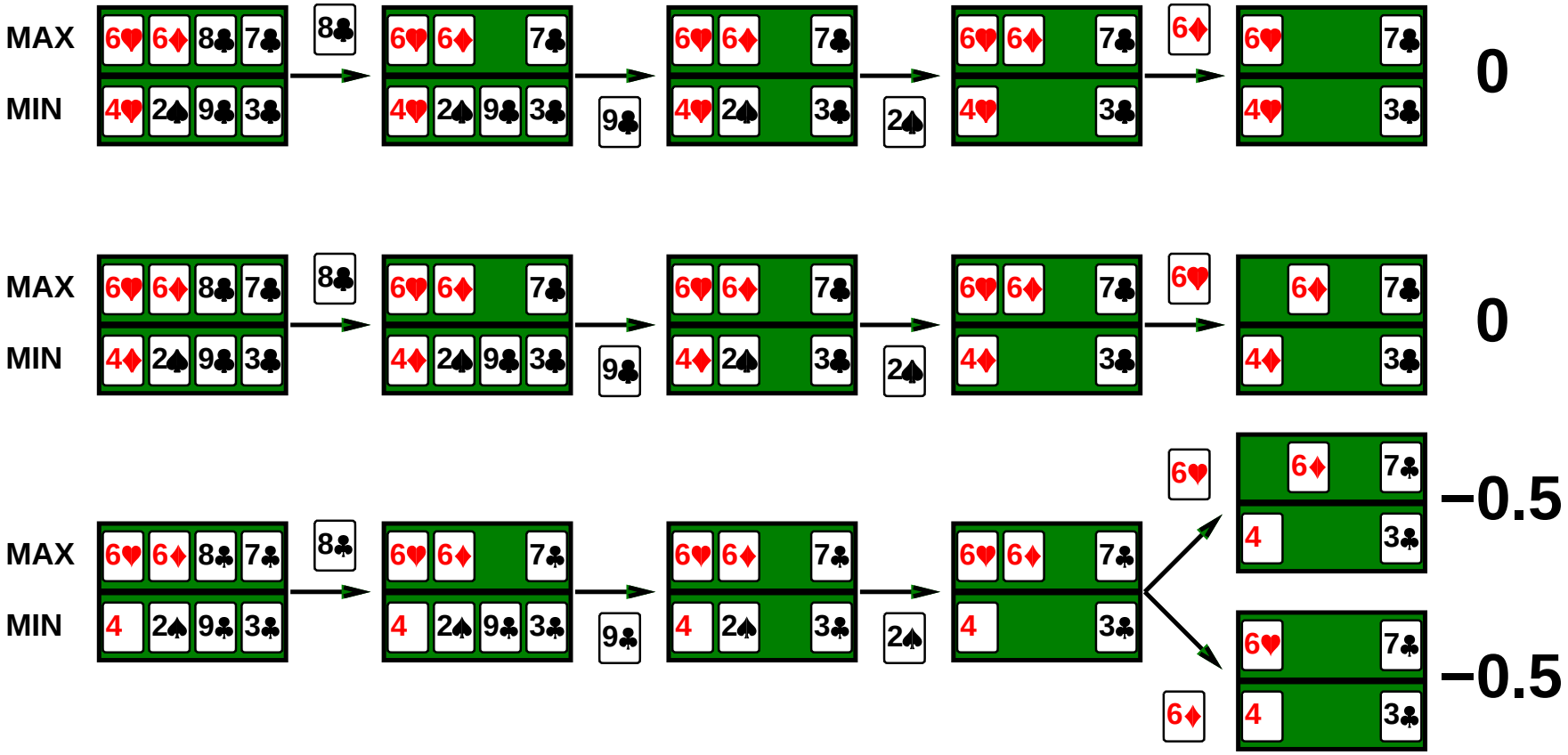
Gry z niepelna informacja: przyklad

Gra w otwarte karty, MAX gra pierwszy



Jeśli MAX rozpocząłby z 6♡ lub 6♢, miałby gwarantowany remis

Gry z niepelna informacja: przyklad



Jeśli MAX rozpocząłby z 6♡ lub 6♢, miałby gwarantowany remis

$\Rightarrow$ uśrednienie nie jest optymalne!

$\Rightarrow$ ważne, która informacja będzie dostępna w którym momencie gry