Sztuczna inteligencja w grach

dr inż. Piotr Beling

Warszawa, 29 grudnia 2016

Obszar zainteresowań

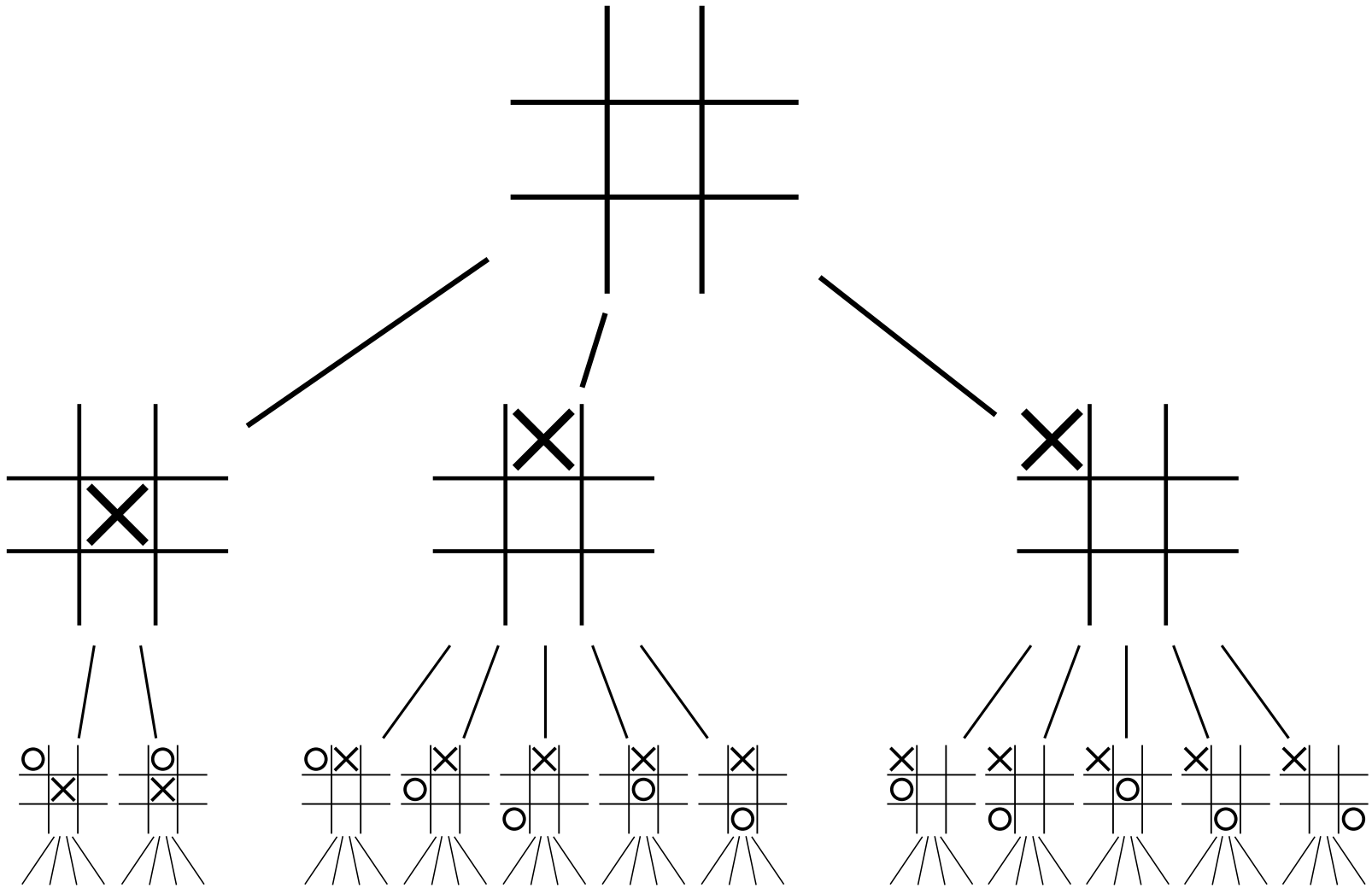
Wyznaczanie suboptymalnych decyzji w grach, głównie:

- dwuosobowych,
- o stałej sumie wypłat,
- deterministycznych,
- skończonych,
- z doskonałą informacją.

Przykłady:

szachy, warcaby, kółko i krzyżyk, GO

Gra jako acykliczny graf skierowany

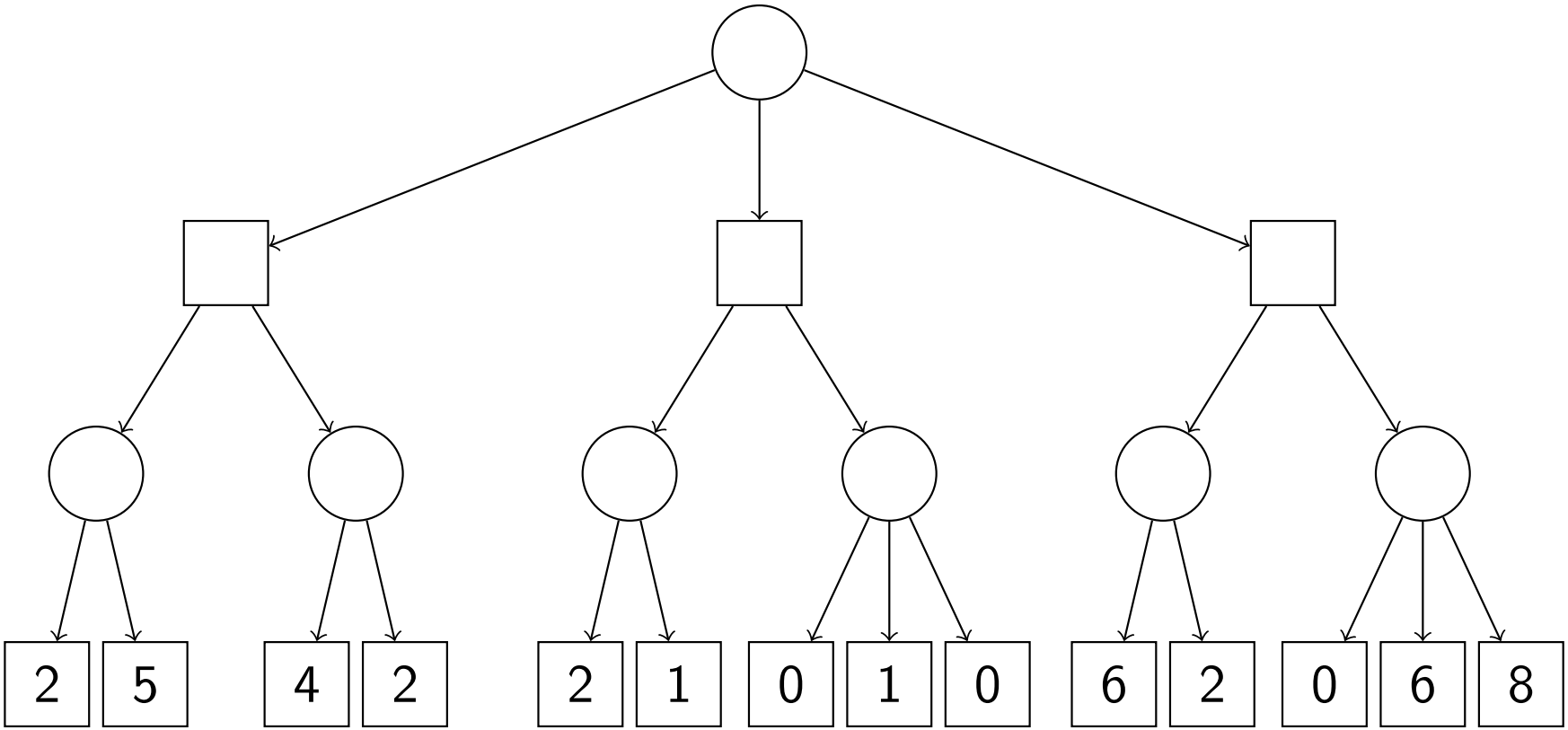


wierzchołki = pozycje gry krawędzie = możliwe ruchy

dla pozycji końcowych (bez następników) określona wypłata graczy

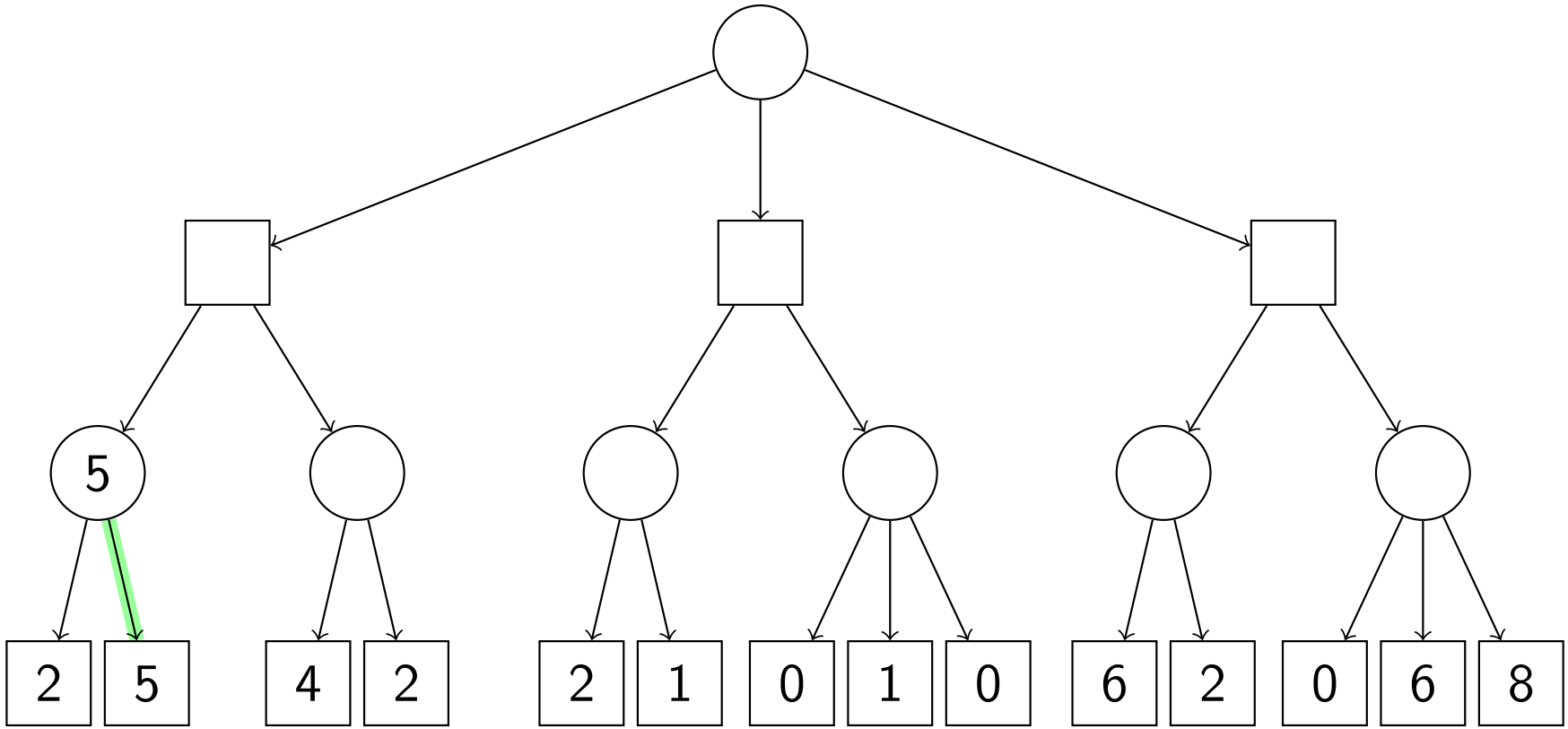
(rysunek pochodzi z *Wikipedii* – Tic-tac-toe-game-tree.svg)

Który ruch wykonać? – wartości minimaksowe pozycji



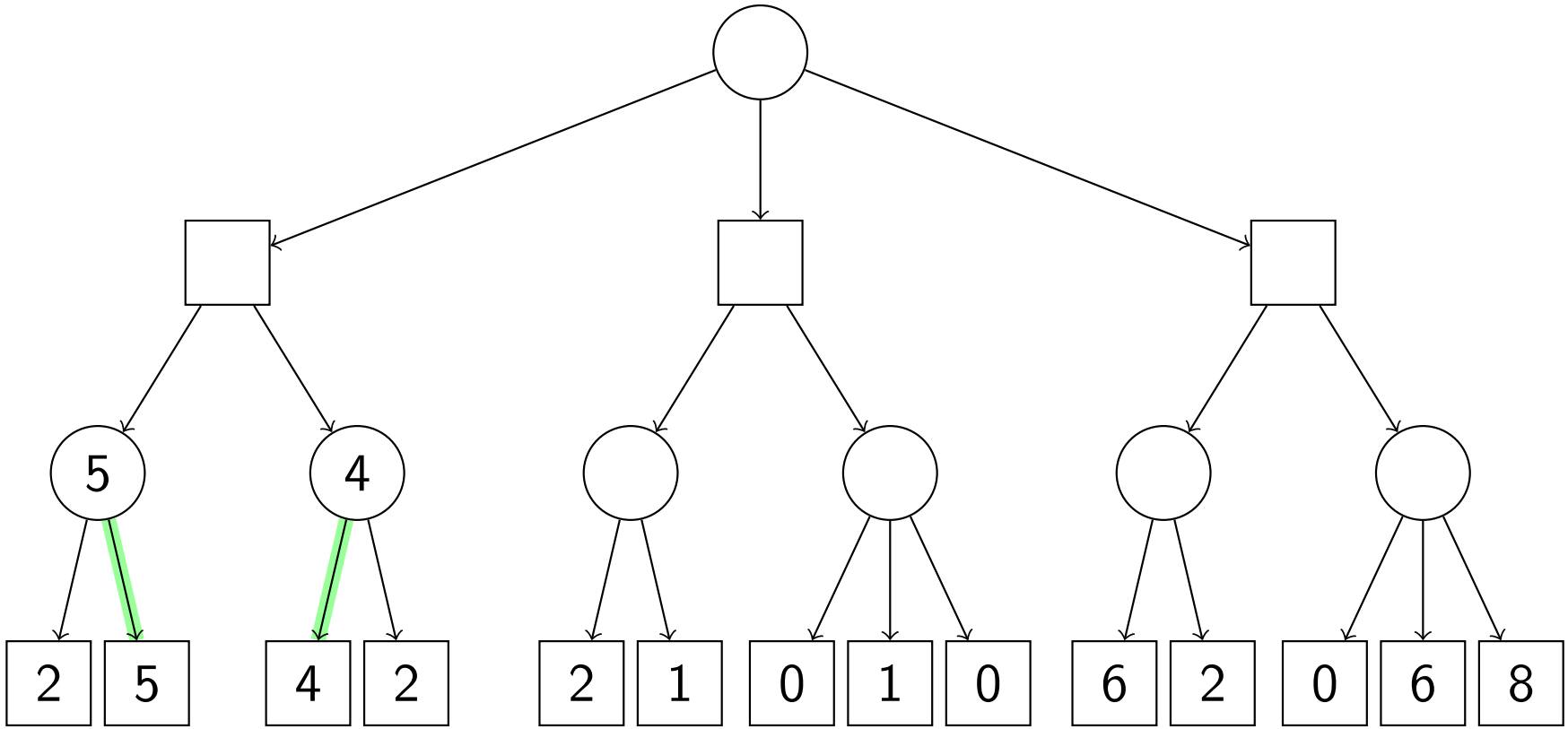
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



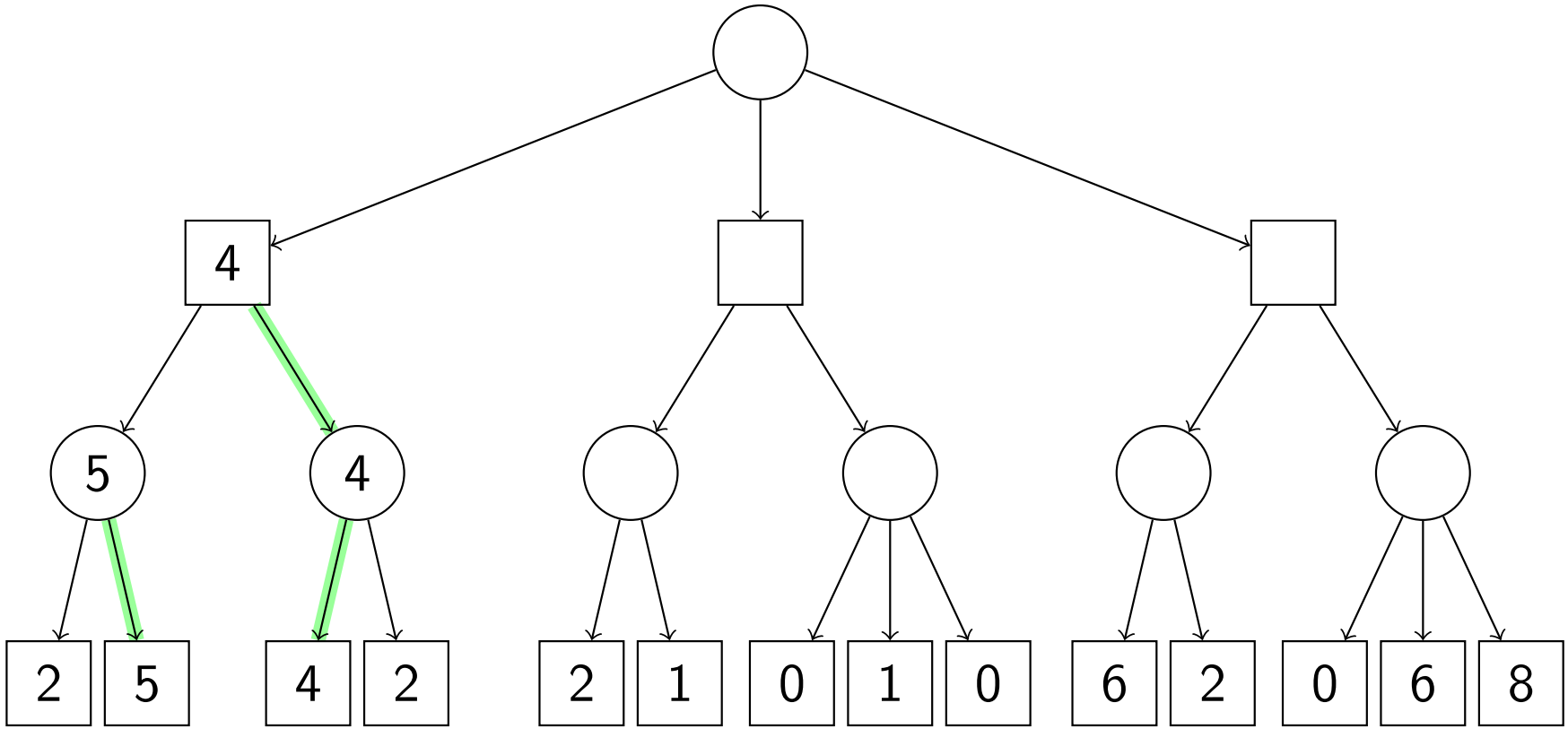
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



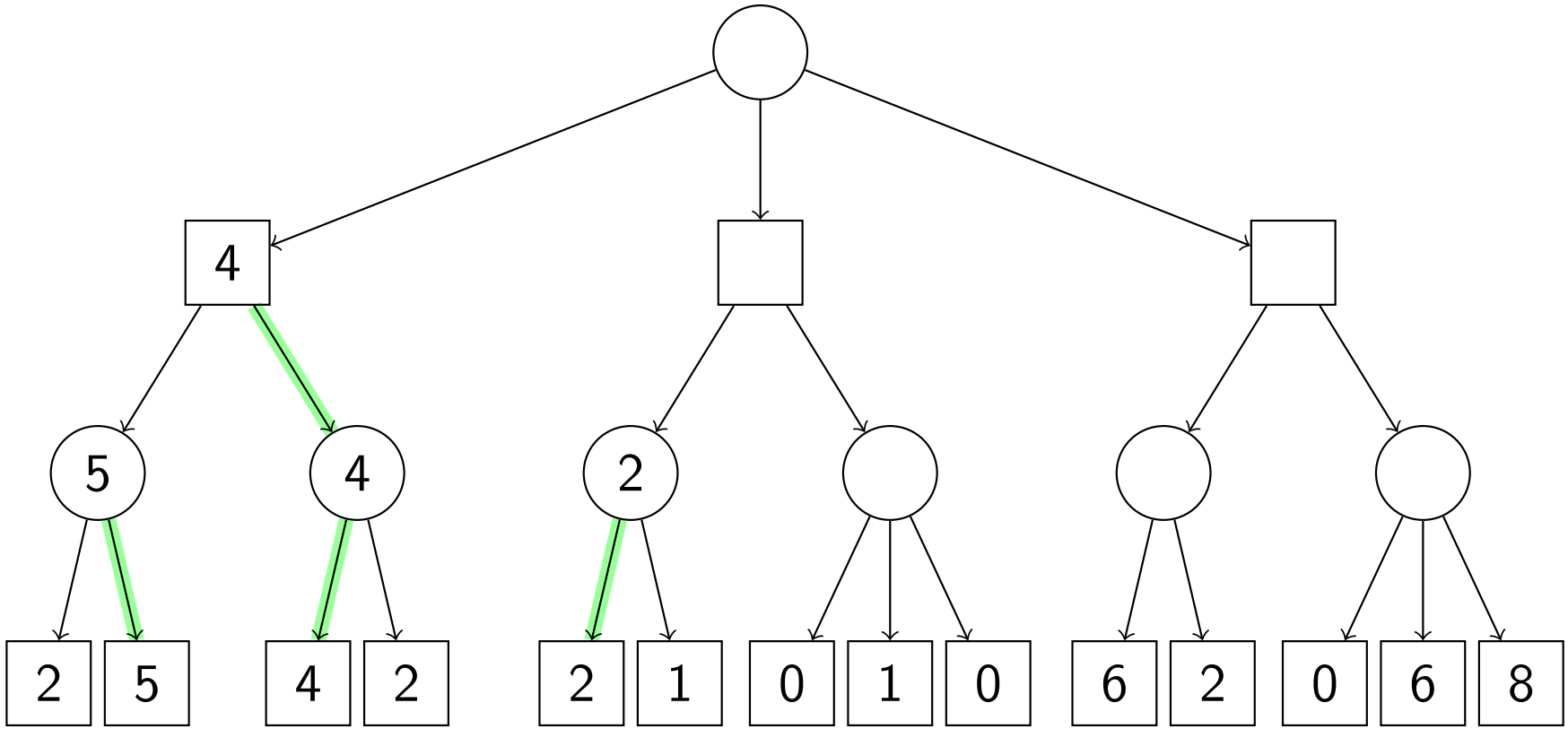
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



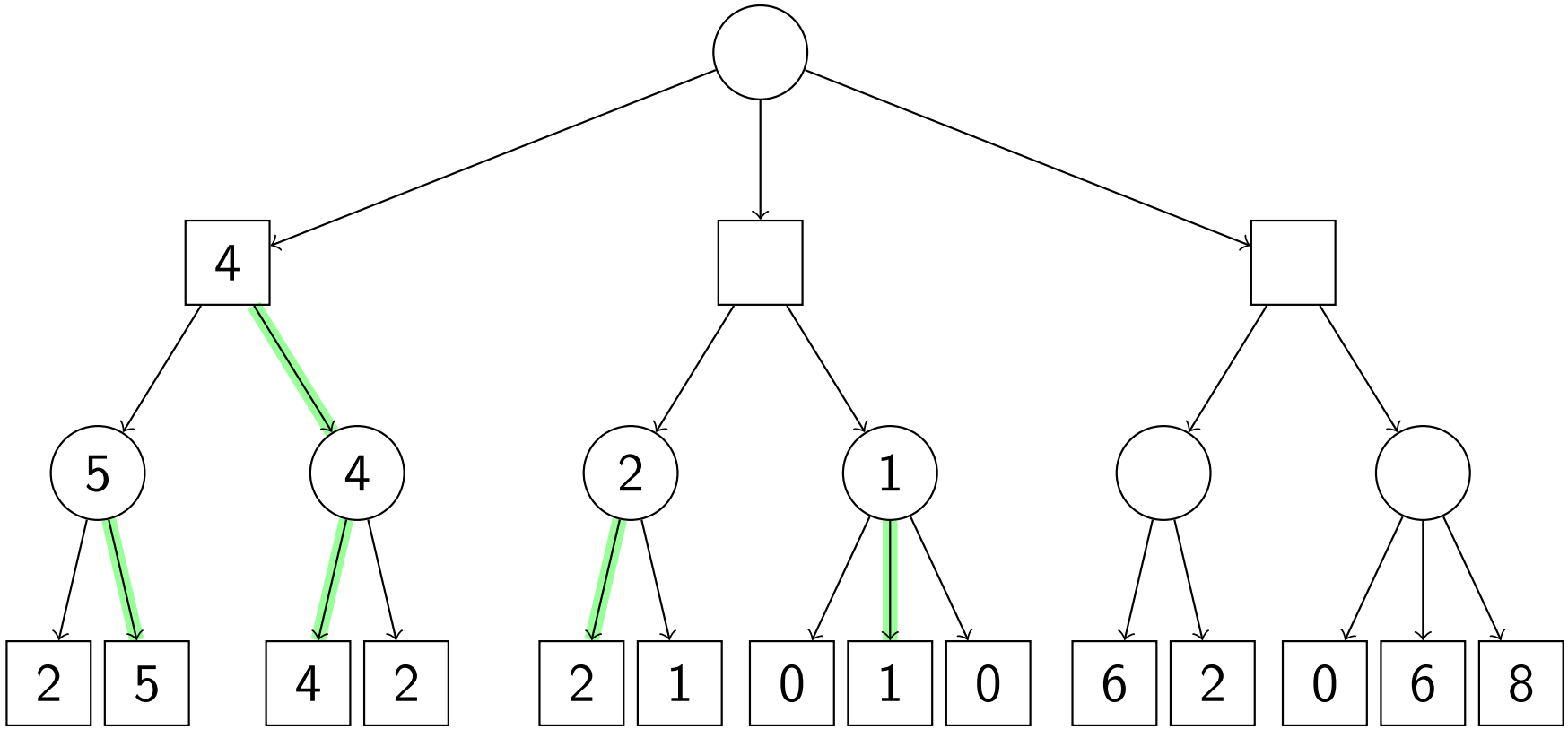
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



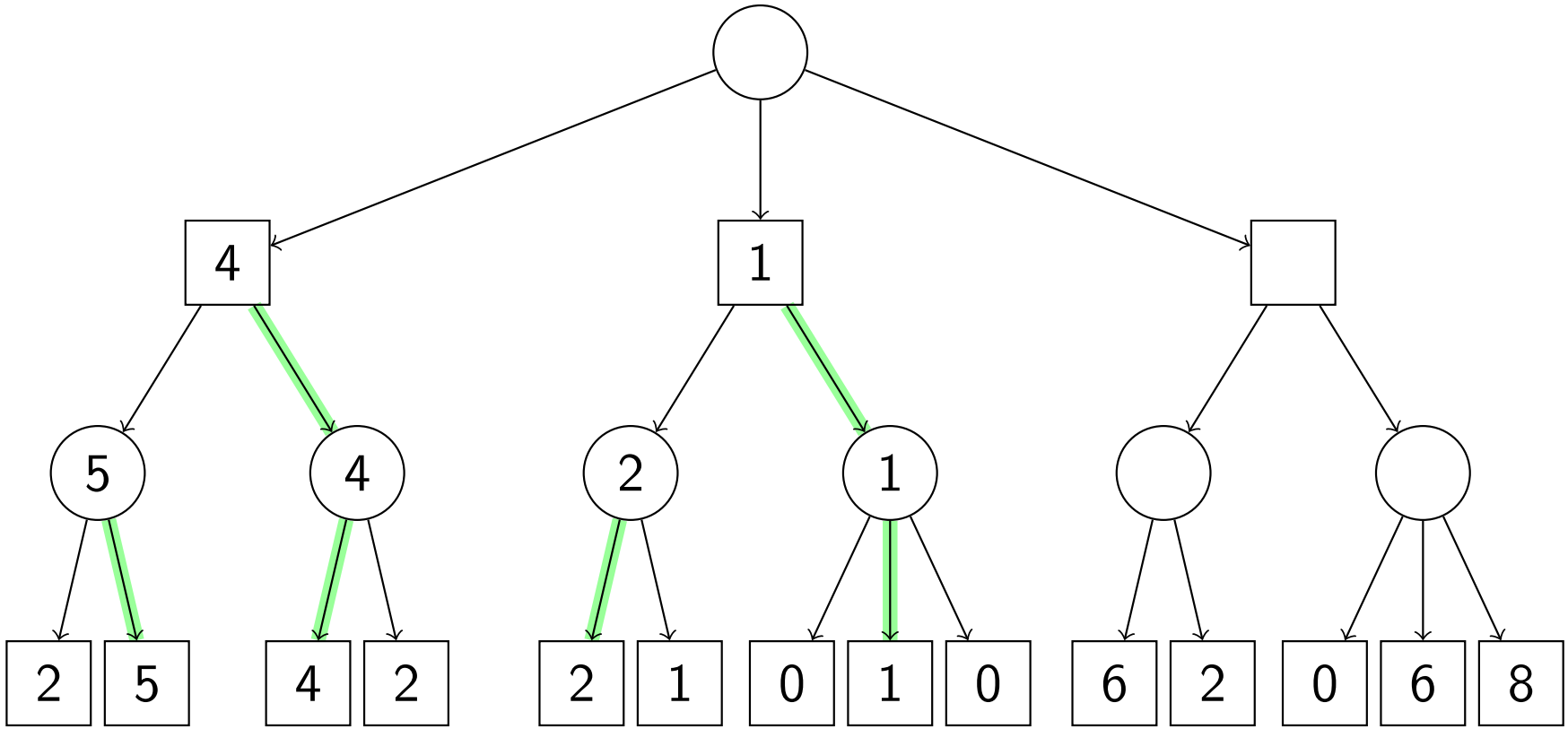
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



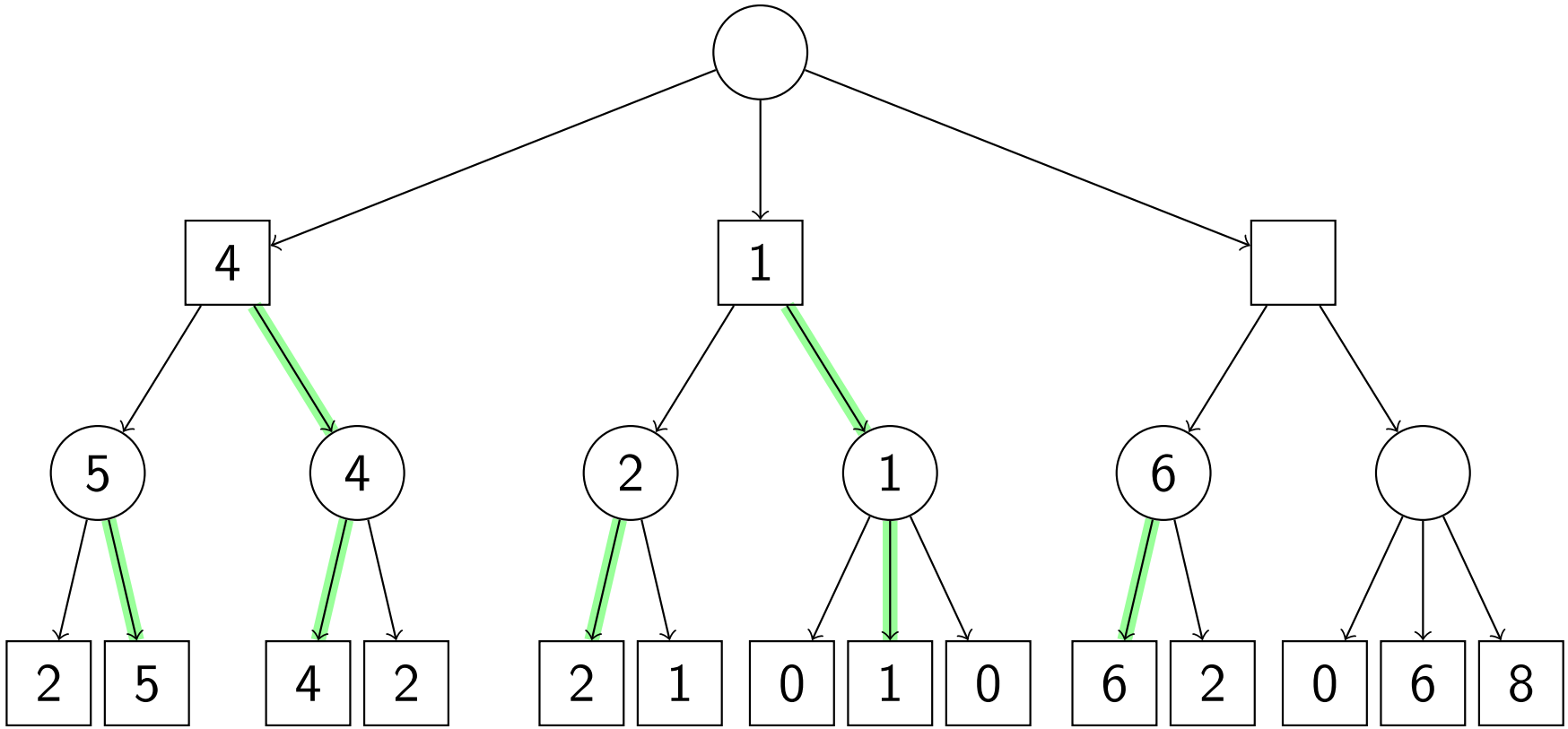
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



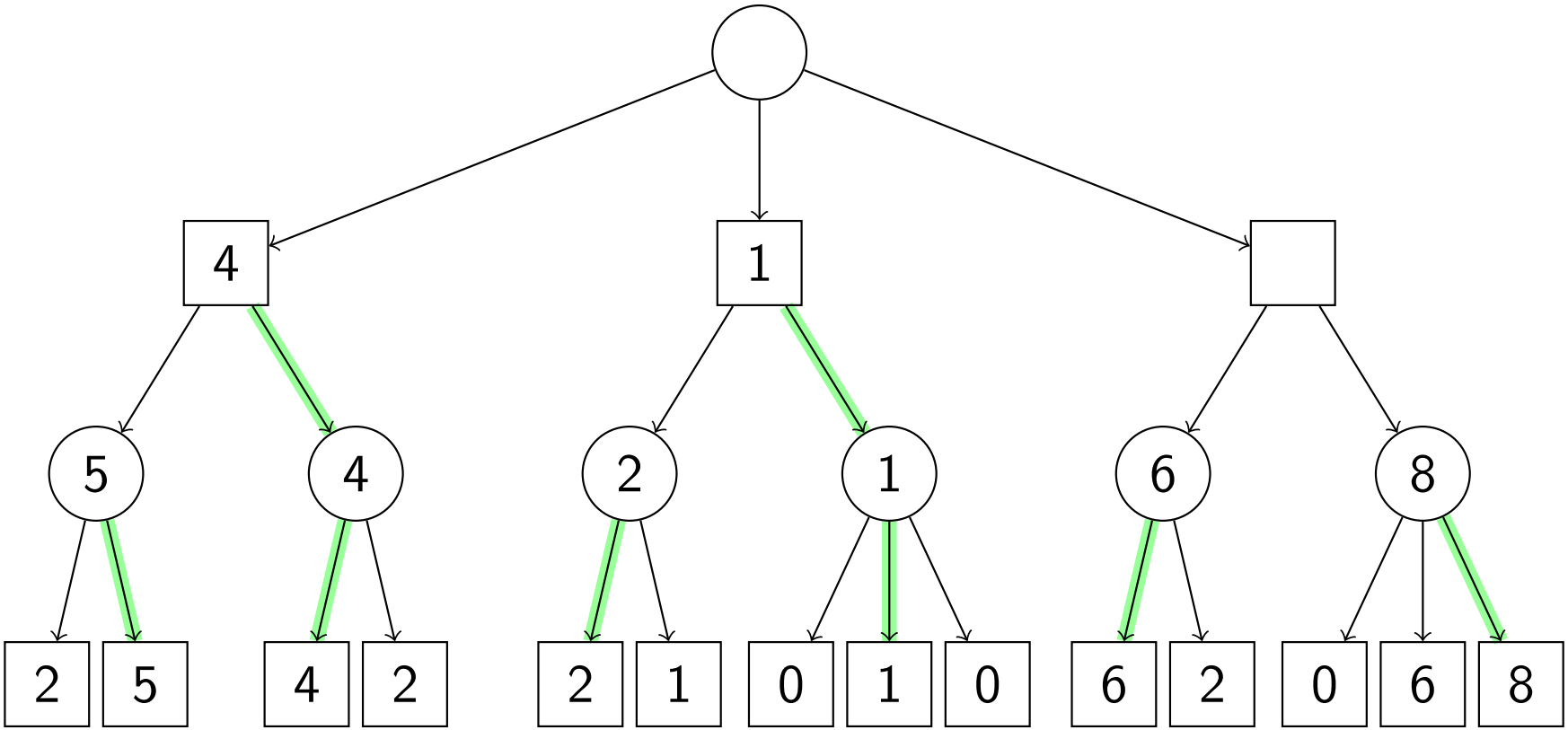
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



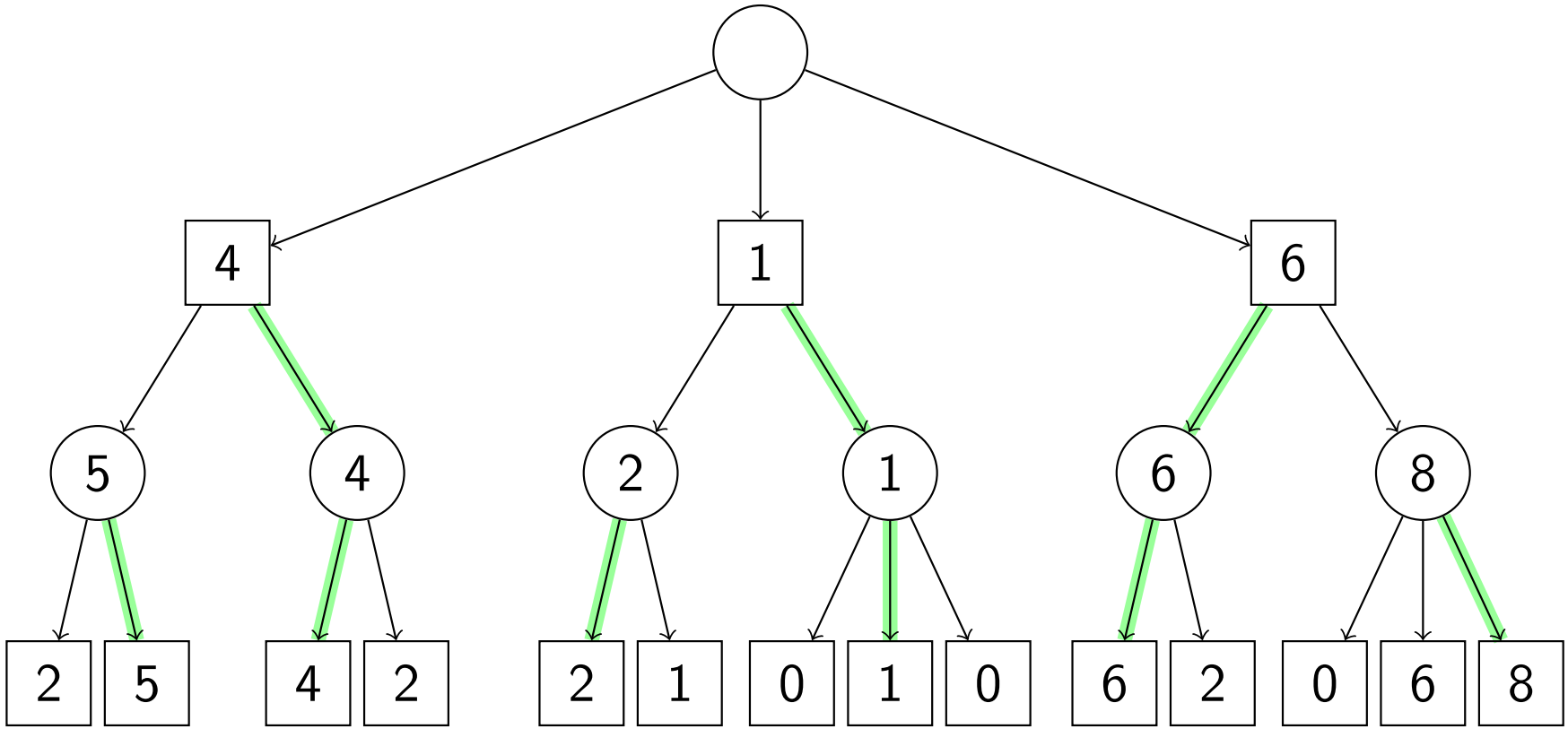
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



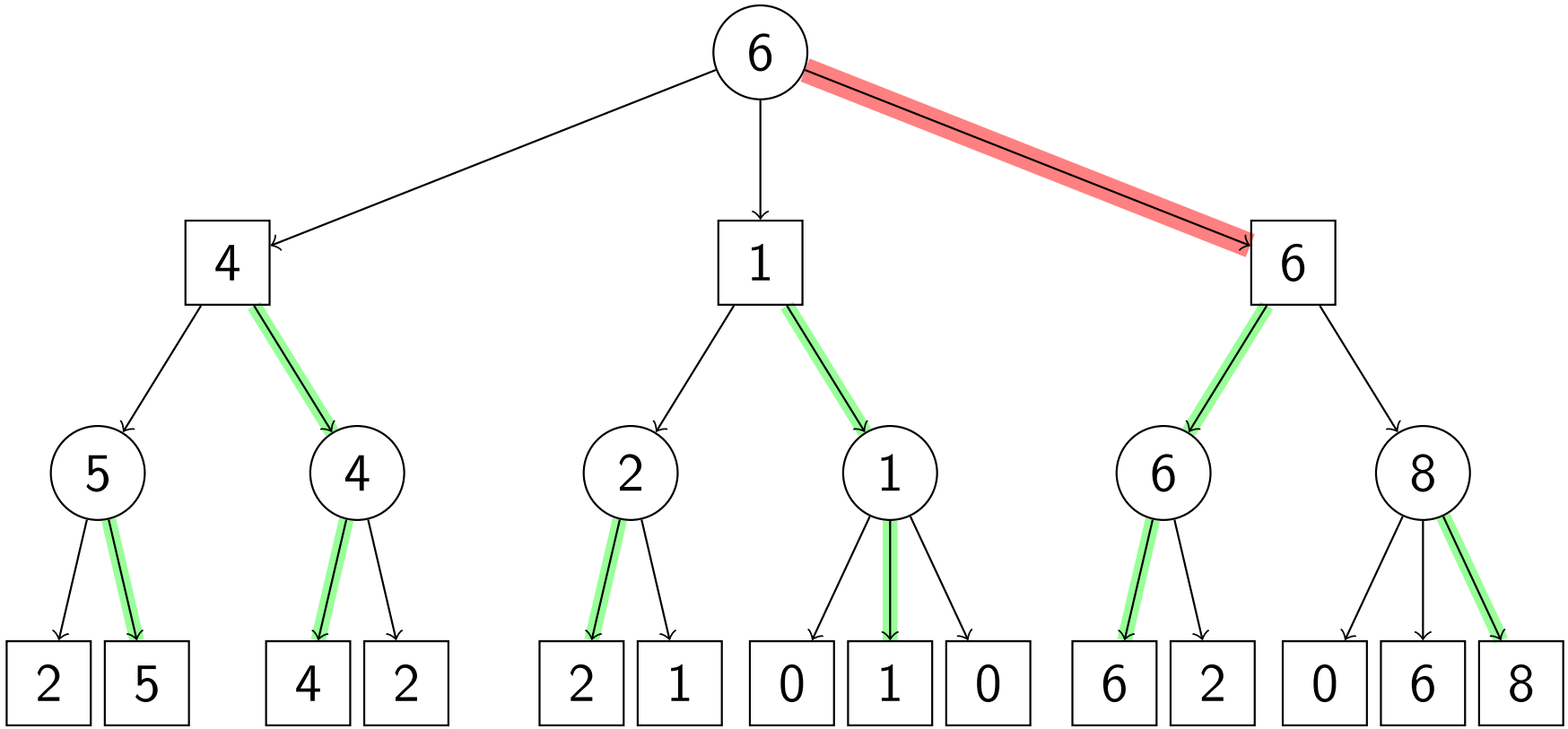
○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



○ – ruch gracza *Max* □ – ruch gracza *Min*

Który ruch wykonać? – wartości minimaksowe pozycji



○ – ruch gracza *Max* □ – ruch gracza *Min*

Szacowana złożoność wybranych gier

gra	logarytm dziesiętny ze złożoności przestrzeni stanów[1]	drzewa gry[2]
Kółko i krzyżyk (3x3)	3	5
Czwórki (Connect Four, 7x6)	13	21
Rozgrywka w widne w brydżu (średnia dla 13 lew)	17	29
Warcaby amerykańskie	18	31
Othello (8x8)	28	58
Szachy	47	123
Go (19x19)	171	360

Źródła: [Allis 1994], [Herik 2002], [Beling doktorat]

W przypadku wielu gier, czas pracy algorytmu z poprzedniego slajdu jest nieakceptowalny.

[1] liczba różnych pozycji osiągalnych z pozycji początkowej

[2] liczba liści w drzewie minimaksowym o głębokości ograniczonej do minimum niezbędnego do ustalenia wartości pozycji początkowej

Szacowana złożoność wybranych gier

gra	logarytm dziesiętny ze złożoności przestrzeni stanów[1]	drzewa gry[2]
Kółko i krzyżyk (3x3)	3	5
Czwórki (Connect Four, 7x6)	13	21
Rozgrywka w widne w brydżu (średnia dla 13 lew)	17	29
Warcaby amerykańskie	18	31
Othello (8x8)	28	58
Szachy	47	123
Go (19x19)	171	360

Źródła: [Allis 1994], [Herik 2002], [Beling doktorat]

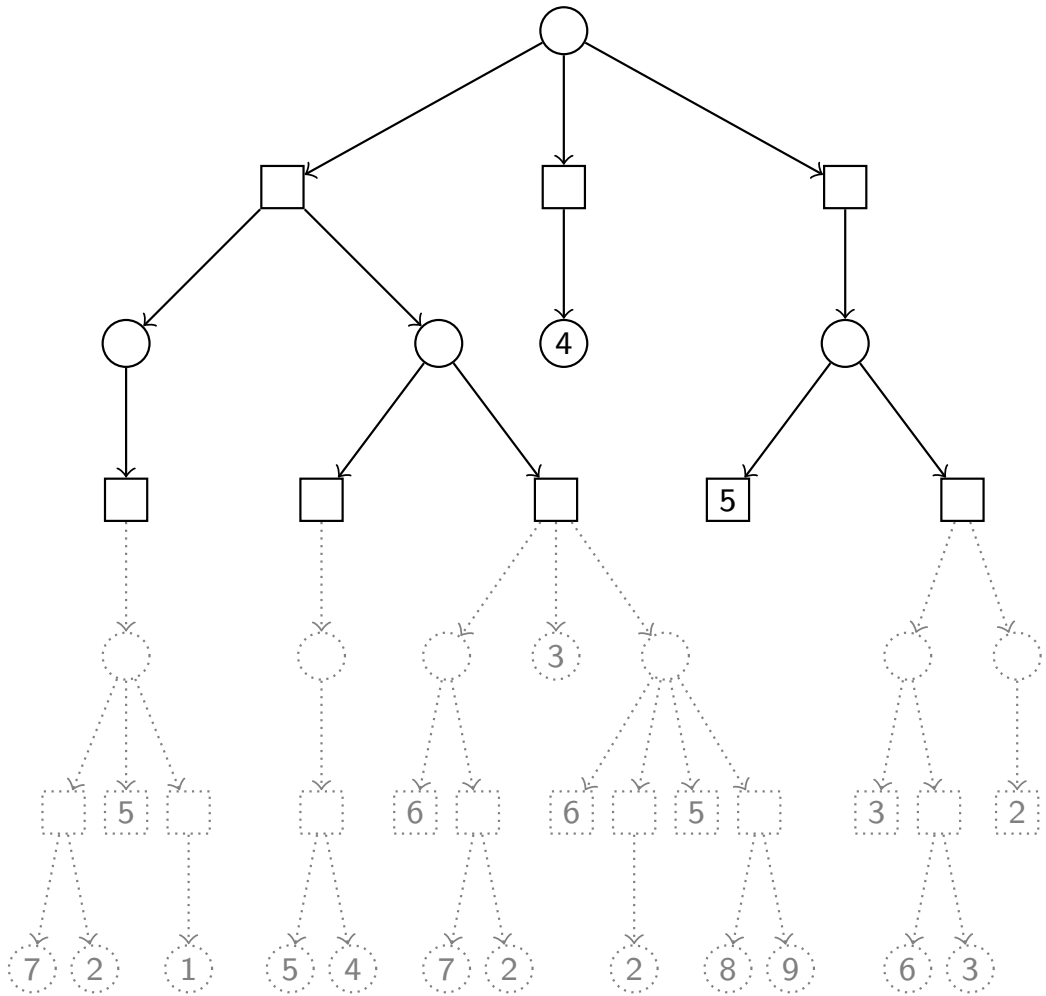
W przypadku wielu gier, czas pracy algorytmu z poprzedniego slajdu jest nieakceptowalny.

[1] liczba różnych pozycji osiągalnych z pozycji początkowej

[2] liczba liści w drzewie minimaksowym o głębokości ograniczonej do minimum niezbędnego do ustalenia wartości pozycji początkowej

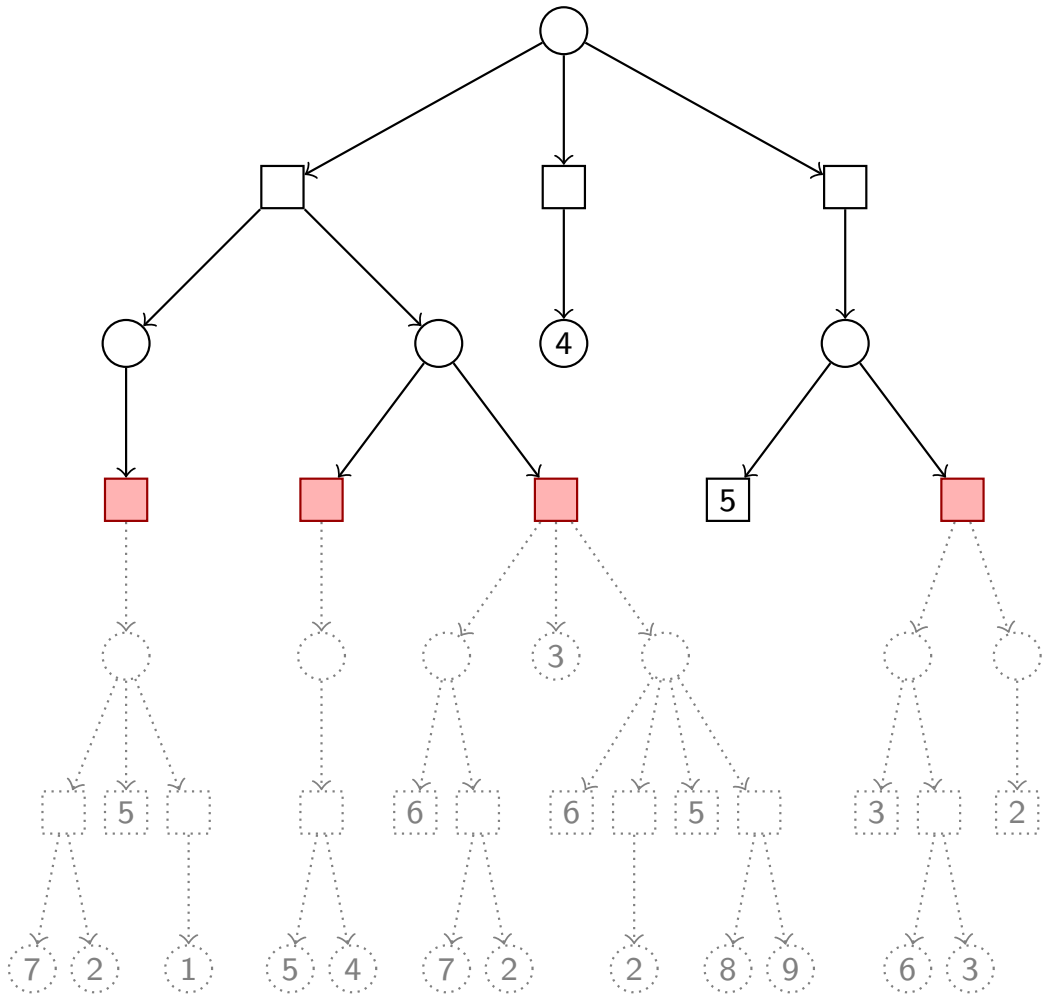
Ograniczenie głębokości poszukiwań

- drzewo poszukiwań ograniczone do zadanej głębokości
- po jej osiągnięciu wartość pozycji jest szacowana heurystycznie (przy wykorzystaniu wiedzy dziedzinowej)
- siła gry rośnie wraz z trafnością tego szacowania oraz głębokością poszukiwań



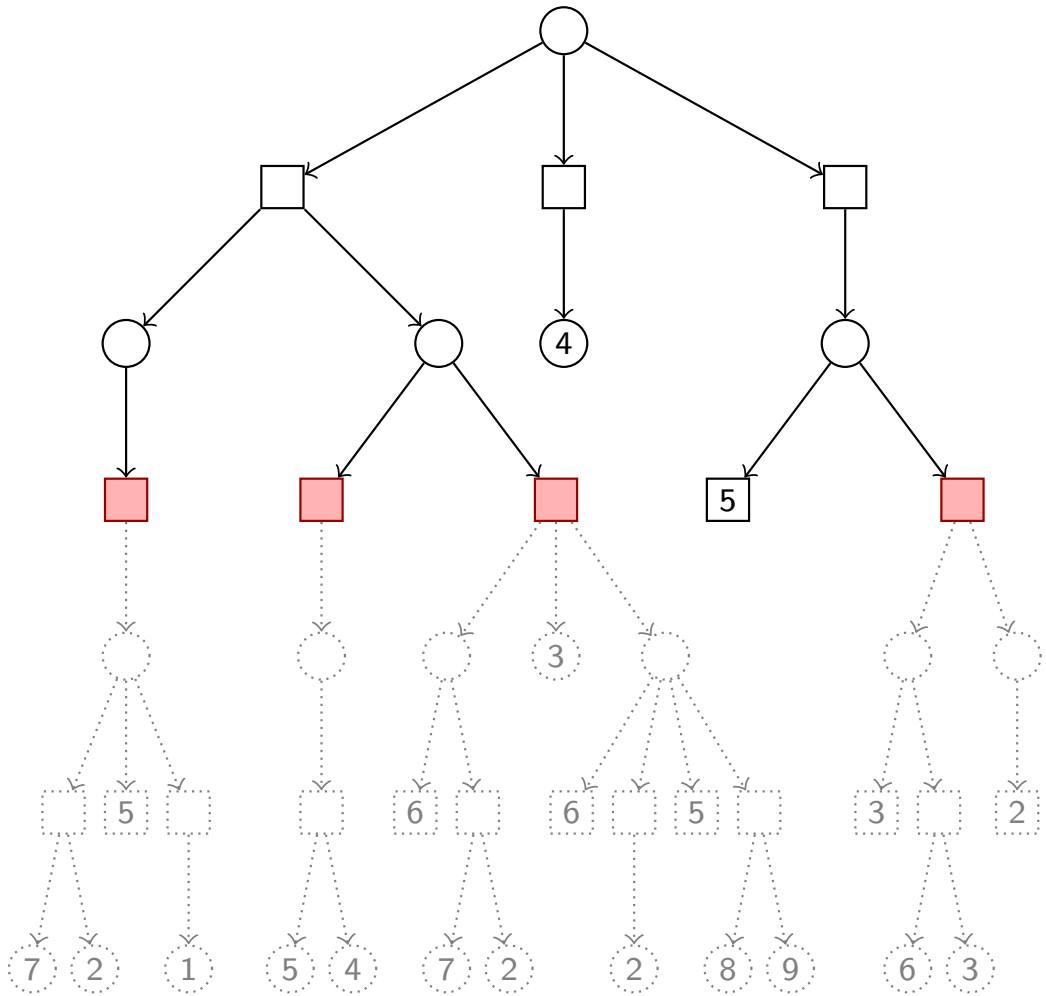
Ograniczenie głębokości poszukiwań

- drzewo poszukiwań ograniczone do zadanej głębokości
- po jej osiągnięciu wartość pozycji jest szacowana heurystycznie (przy wykorzystaniu wiedzy dziedzinowej)
- siła gry rośnie wraz z trafnością tego szacowania oraz głębokością poszukiwań



Ograniczenie głębokości poszukiwań

- drzewo poszukiwań ograniczone do zadanej głębokości
- po jej osiągnięciu wartość pozycji jest szacowana heurystycznie (przy wykorzystaniu wiedzy dziedzinowej)
- siła gry rośnie wraz z trafnością tego szacowania oraz głębokością poszukiwań



Algorytm α-β [Knuth 1975] – idea

- załóżmy, że w pozycji P ruch należy do gracza *Max*
- i wiadomo, że wartość jednego z następników P wynosi α;
- zatem wartość P to co najmniej α
- i do jej wyznaczenia nie potrzebujemy znać dokładnych wartość następników o wartościach nie większych od α;
- analogicznie, jeżeli w pozycji P ruch należy do gracza *Min*, to nie potrzebujemy znać dokładnych wartość następników P nie mniejszych od dotychczas znalezionego minimum β;
- w ten sposób otrzymujemy i następnie, idąc w głąb drzewa poszukiwań, zawężamy przedział (zwany też „oknem") wartości (α, β);
- przeszukiwanie węzła można przerwać gdy tylko zorientujemy się, że jego wartość nie mieści się w oknie (α, β).

Algorytm α-β [Knuth 1975] – idea

- załóżmy, że w pozycji P ruch należy do gracza *Max*
- i wiadomo, że wartość jednego z następników P wynosi α;
- zatem wartość P to co najmniej α
- i do jej wyznaczenia nie potrzebujemy znać dokładnych wartość następników o wartościach nie większych od α;
- analogicznie, jeżeli w pozycji P ruch należy do gracza *Min*, to nie potrzebujemy znać dokładnych wartość następników P nie mniejszych od dotychczas znalezionego minimum β;
- w ten sposób otrzymujemy i następnie, idąc w głąb drzewa poszukiwań, zawężamy przedział (zwany też „oknem") wartości (α, β);
- przeszukiwanie węzła można przerwać gdy tylko zorientujemy się, że jego wartość nie mieści się w oknie (α, β).

Algorytm α-β [Knuth 1975] – idea

- załóżmy, że w pozycji P ruch należy do gracza *Max*
- i wiadomo, że wartość jednego z następników P wynosi α;
- zatem wartość P to co najmniej α
- i do jej wyznaczenia nie potrzebujemy znać dokładnych wartość następników o wartościach nie większych od α;
- analogicznie, jeżeli w pozycji P ruch należy do gracza *Min*, to nie potrzebujemy znać dokładnych wartość następników P nie mniejszych od dotychczas znalezionego minimum β;
- w ten sposób otrzymujemy i następnie, idąc w głąb drzewa poszukiwań, zawężamy przedział (zwany też „oknem") wartości (α, β);
- przeszukiwanie węzła można przerwać gdy tylko zorientujemy się, że jego wartość nie mieści się w oknie (α, β).

Algorytm α-β [Knuth 1975] – idea

- załóżmy, że w pozycji P ruch należy do gracza *Max*
- i wiadomo, że wartość jednego z następników P wynosi α;
- zatem wartość P to co najmniej α
- i do jej wyznaczenia nie potrzebujemy znać dokładnych wartość następników o wartościach nie większych od α;
- analogicznie, jeżeli w pozycji P ruch należy do gracza *Min*, to nie potrzebujemy znać dokładnych wartość następników P nie mniejszych od dotychczas znalezionego minimum β;
- w ten sposób otrzymujemy i następnie, idąc w głąb drzewa poszukiwań, zawężamy przedział (zwany też „oknem”) wartości (α, β);
- przeszukiwanie węzła można przerwać gdy tylko zorientujemy się, że jego wartość nie mieści się w oknie (α, β).

Algorytm α-β [Knuth 1975] – idea

- załóżmy, że w pozycji P ruch należy do gracza *Max*
- i wiadomo, że wartość jednego z następników P wynosi α;
- zatem wartość P to co najmniej α
- i do jej wyznaczenia nie potrzebujemy znać dokładnych wartość następników o wartościach nie większych od α;
- analogicznie, jeżeli w pozycji P ruch należy do gracza *Min*, to nie potrzebujemy znać dokładnych wartość następników P nie mniejszych od dotychczas znalezionego minimum β;
- w ten sposób otrzymujemy i następnie, idąc w głąb drzewa poszukiwań, zawężamy przedział (zwany też „oknem") wartości (α, β);
- przeszukiwanie węzła można przerwać gdy tylko zorientujemy się, że jego wartość nie mieści się w oknie (α, β).

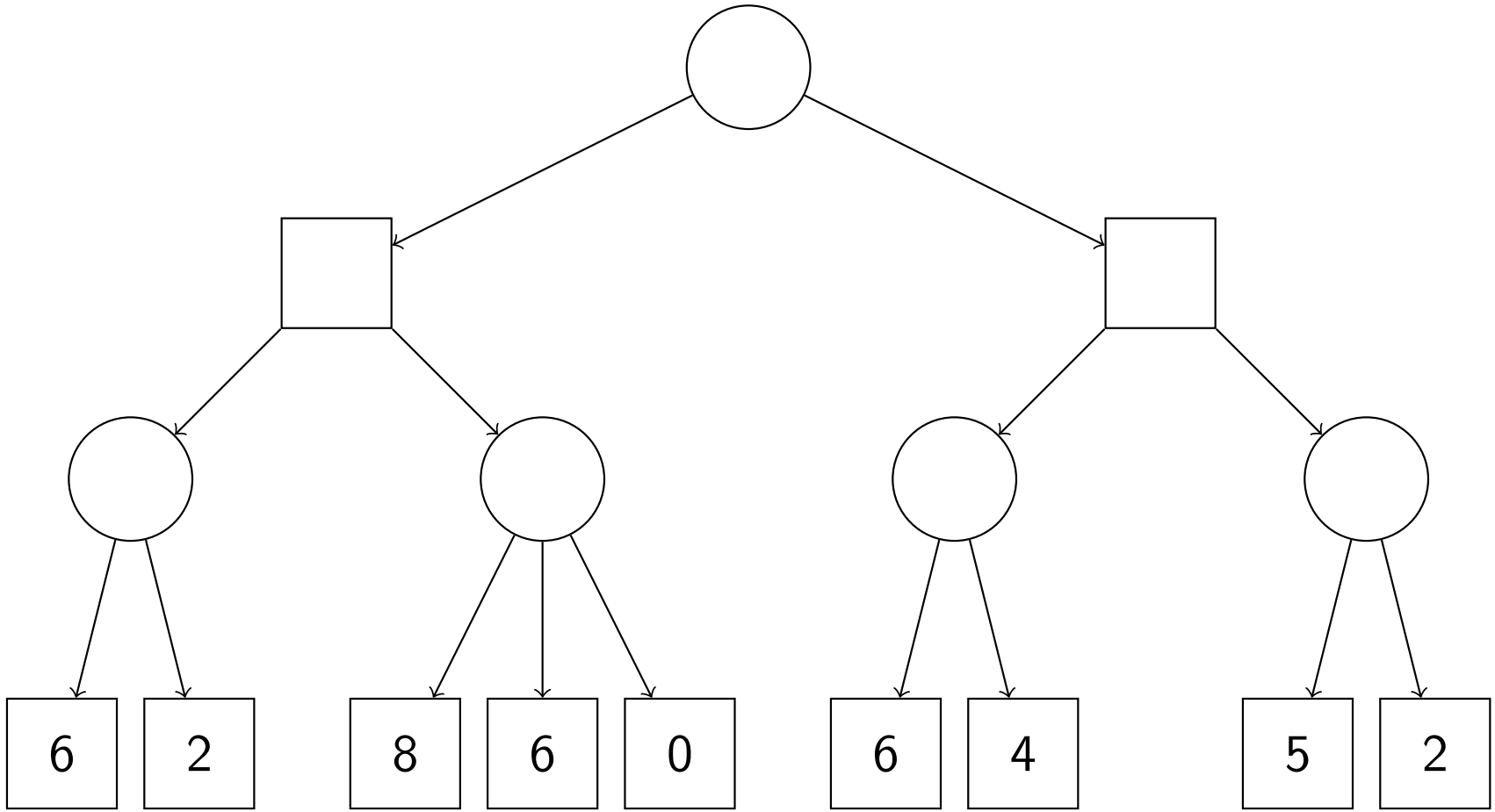
Algorytm α-β [Knuth 1975] – idea

- załóżmy, że w pozycji P ruch należy do gracza *Max*
- i wiadomo, że wartość jednego z następników P wynosi α;
- zatem wartość P to co najmniej α
- i do jej wyznaczenia nie potrzebujemy znać dokładnych wartość następników o wartościach nie większych od α;
- analogicznie, jeżeli w pozycji P ruch należy do gracza *Min*, to nie potrzebujemy znać dokładnych wartość następników P nie mniejszych od dotychczas znalezionego minimum β;
- w ten sposób otrzymujemy i następnie, idąc w głąb drzewa poszukiwań, zawężamy przedział (zwany też „oknem") wartości (α, β);
- przeszukiwanie węzła można przerwać gdy tylko zorientujemy się, że jego wartość nie mieści się w oknie (α, β).

Algorytm α-β [Knuth 1975] – idea

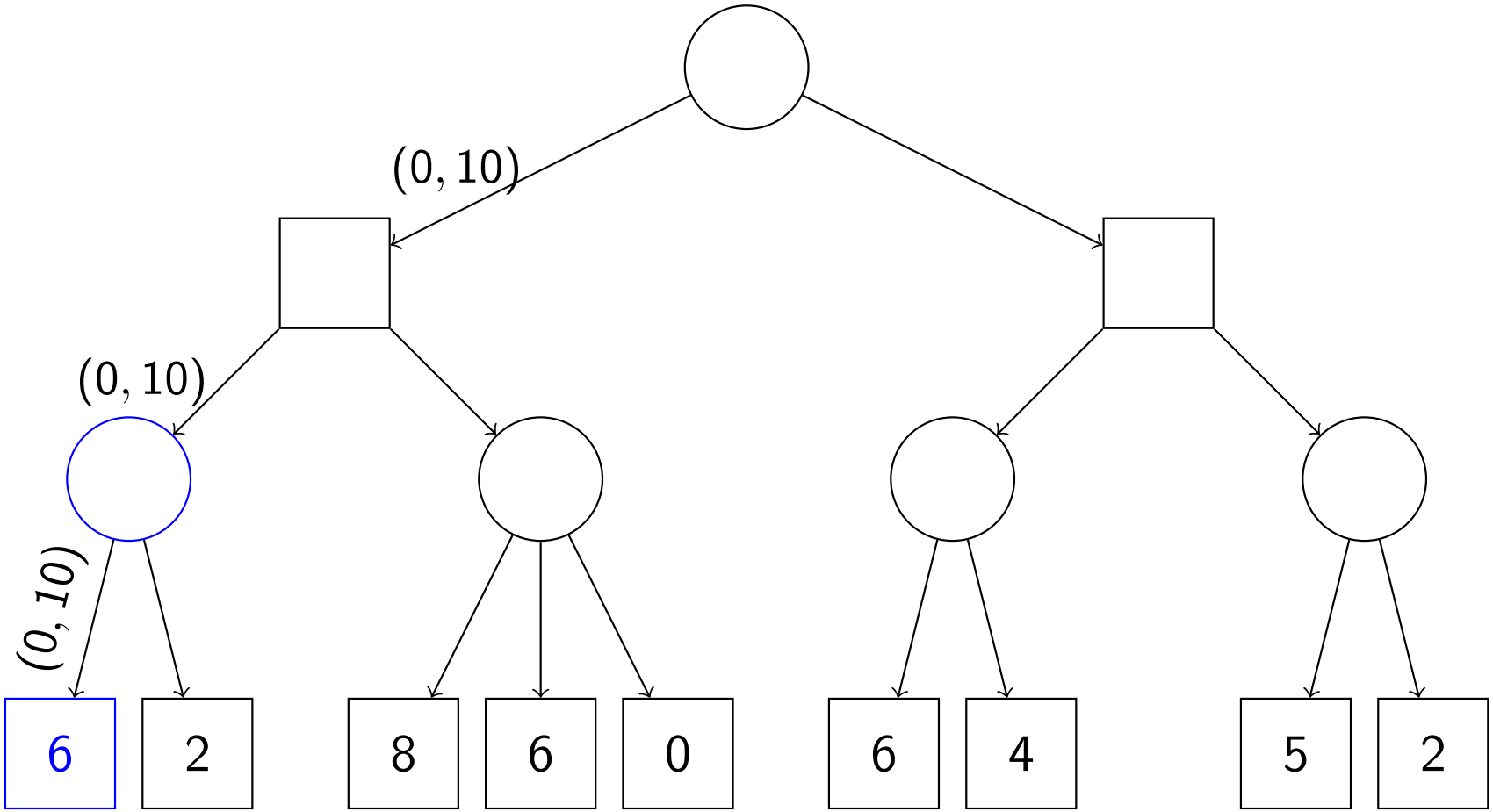
- załóżmy, że w pozycji P ruch należy do gracza *Max*
- i wiadomo, że wartość jednego z następników P wynosi α;
- zatem wartość P to co najmniej α
- i do jej wyznaczenia nie potrzebujemy znać dokładnych wartość następników o wartościach nie większych od α;
- analogicznie, jeżeli w pozycji P ruch należy do gracza *Min*, to nie potrzebujemy znać dokładnych wartość następników P nie mniejszych od dotychczas znalezionego minimum β;
- w ten sposób otrzymujemy i następnie, idąc w głąb drzewa poszukiwań, zawężamy przedział (zwany też „oknem") wartości (α, β);
- przeszukiwanie węzła można przerwać gdy tylko zorientujemy się, że jego wartość nie mieści się w oknie (α, β).

Algorytm α-β [Knuth 1975] – przykład



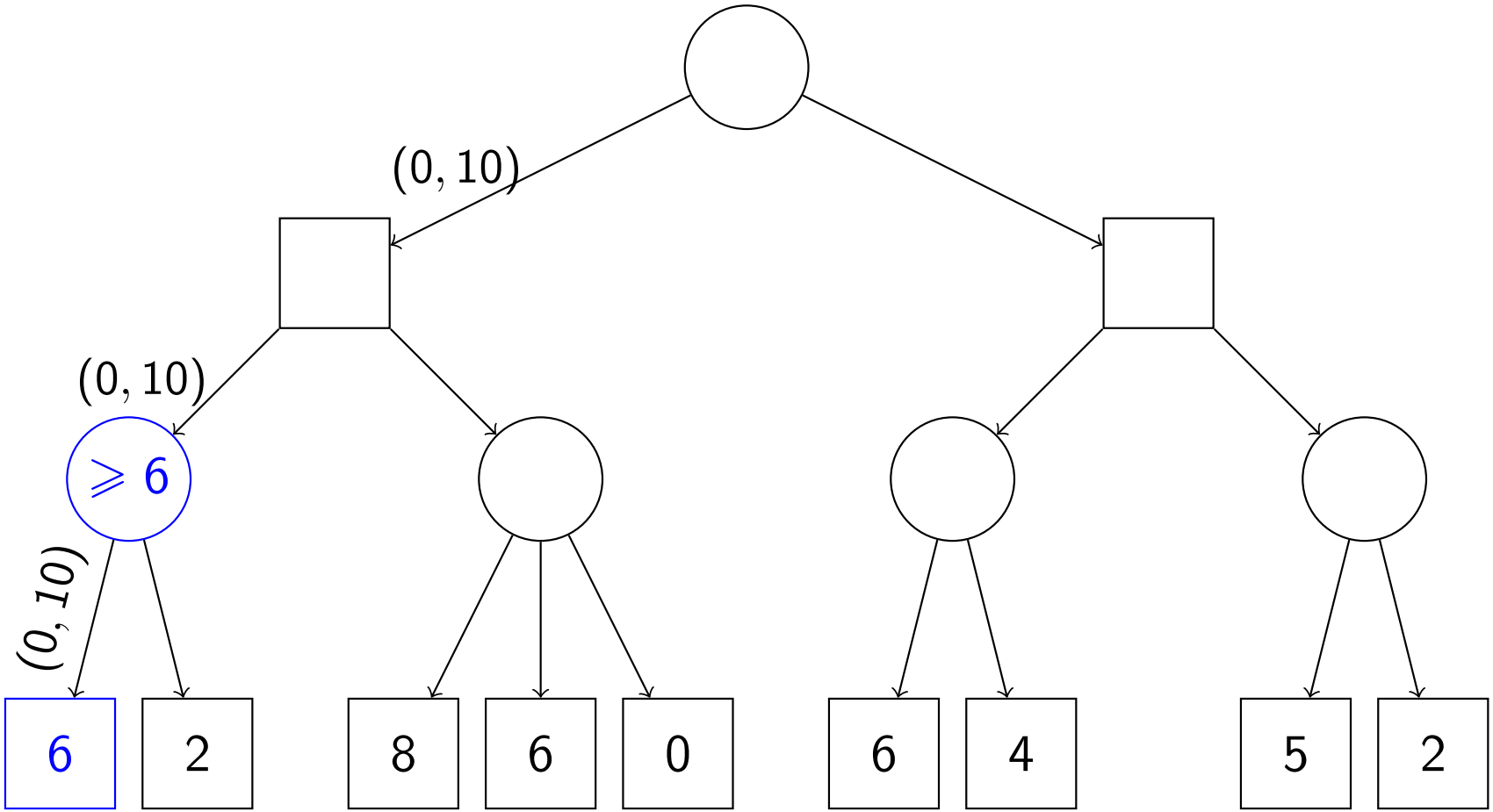
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



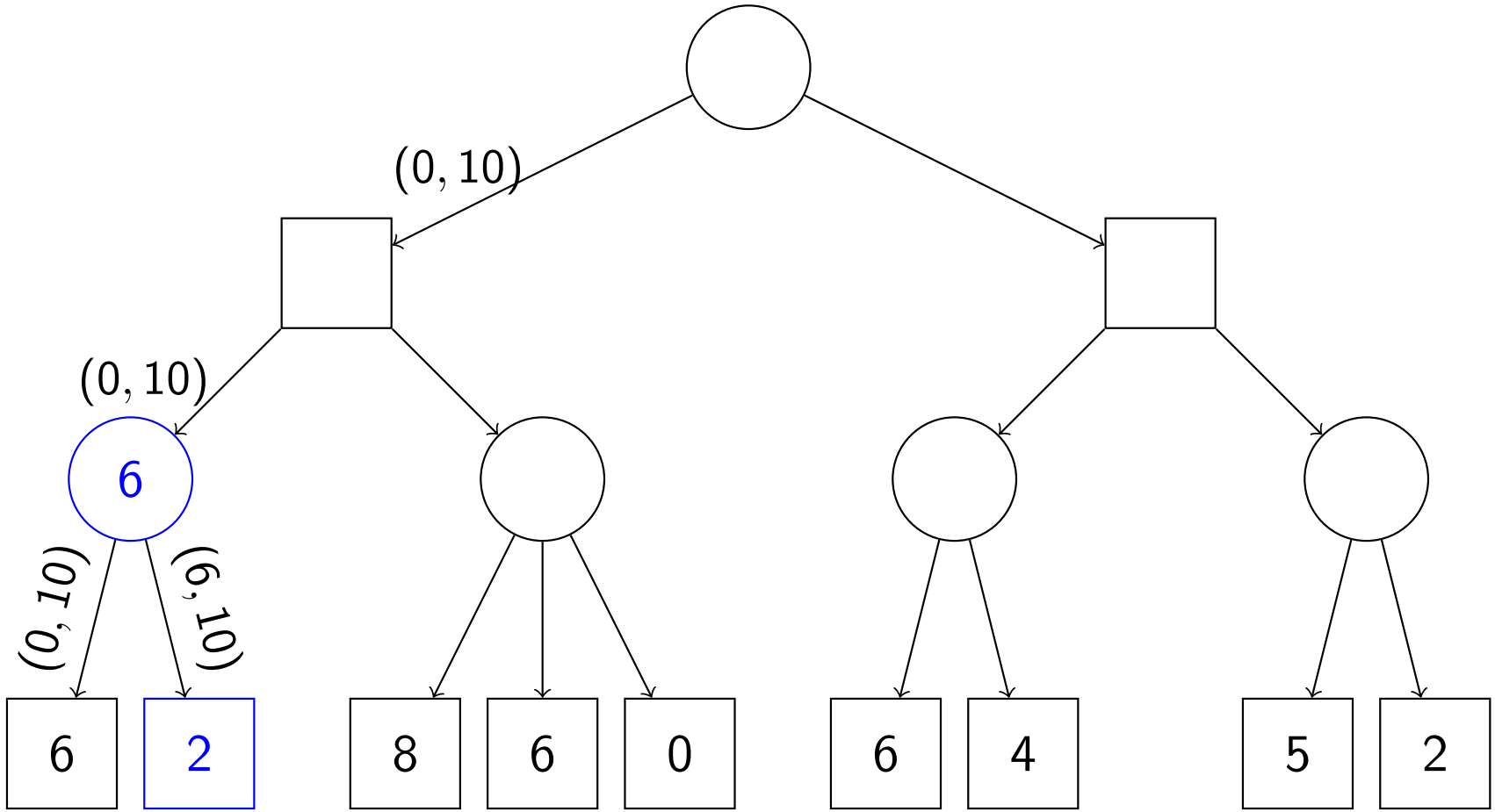
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



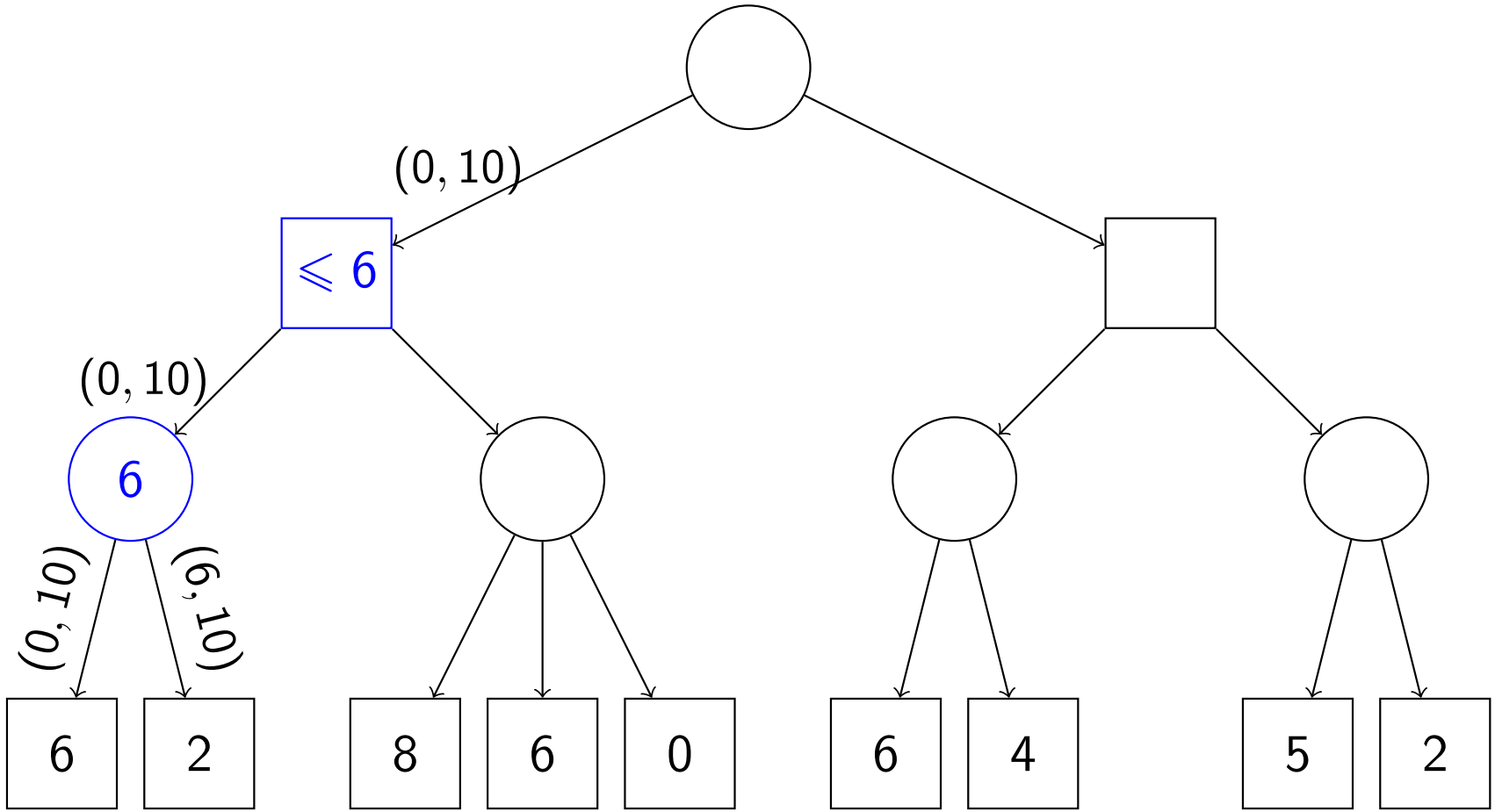
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



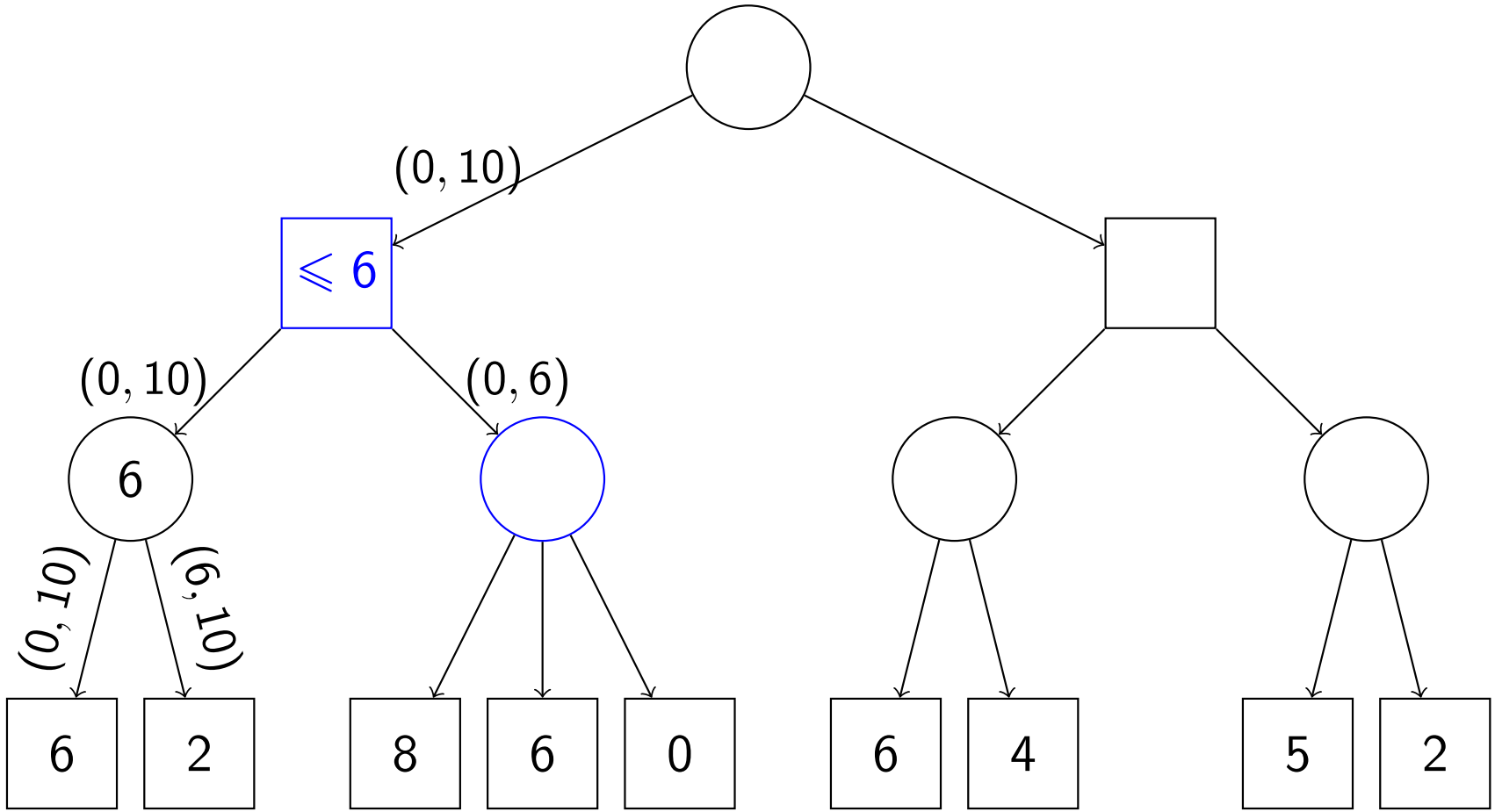
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



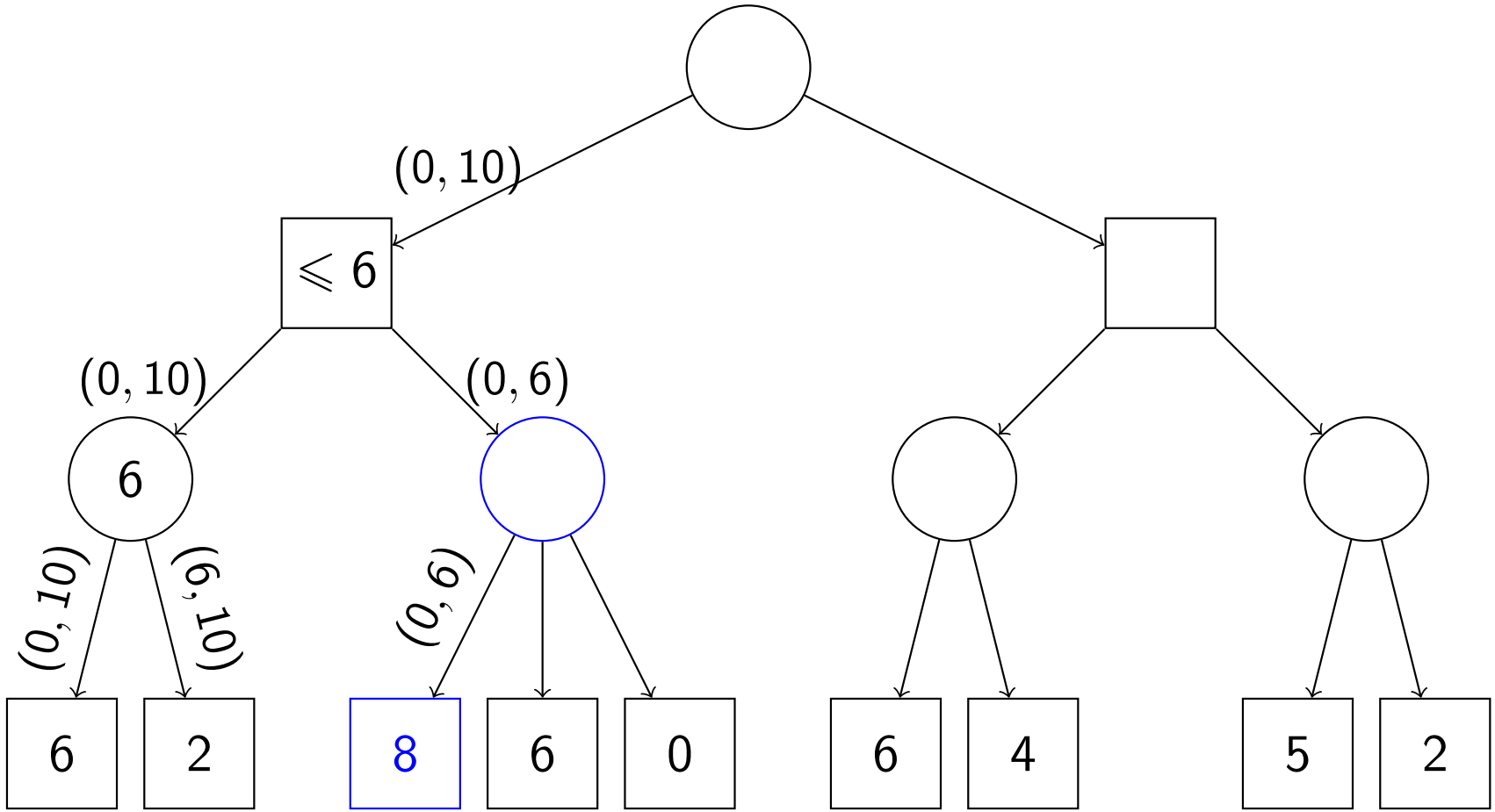
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



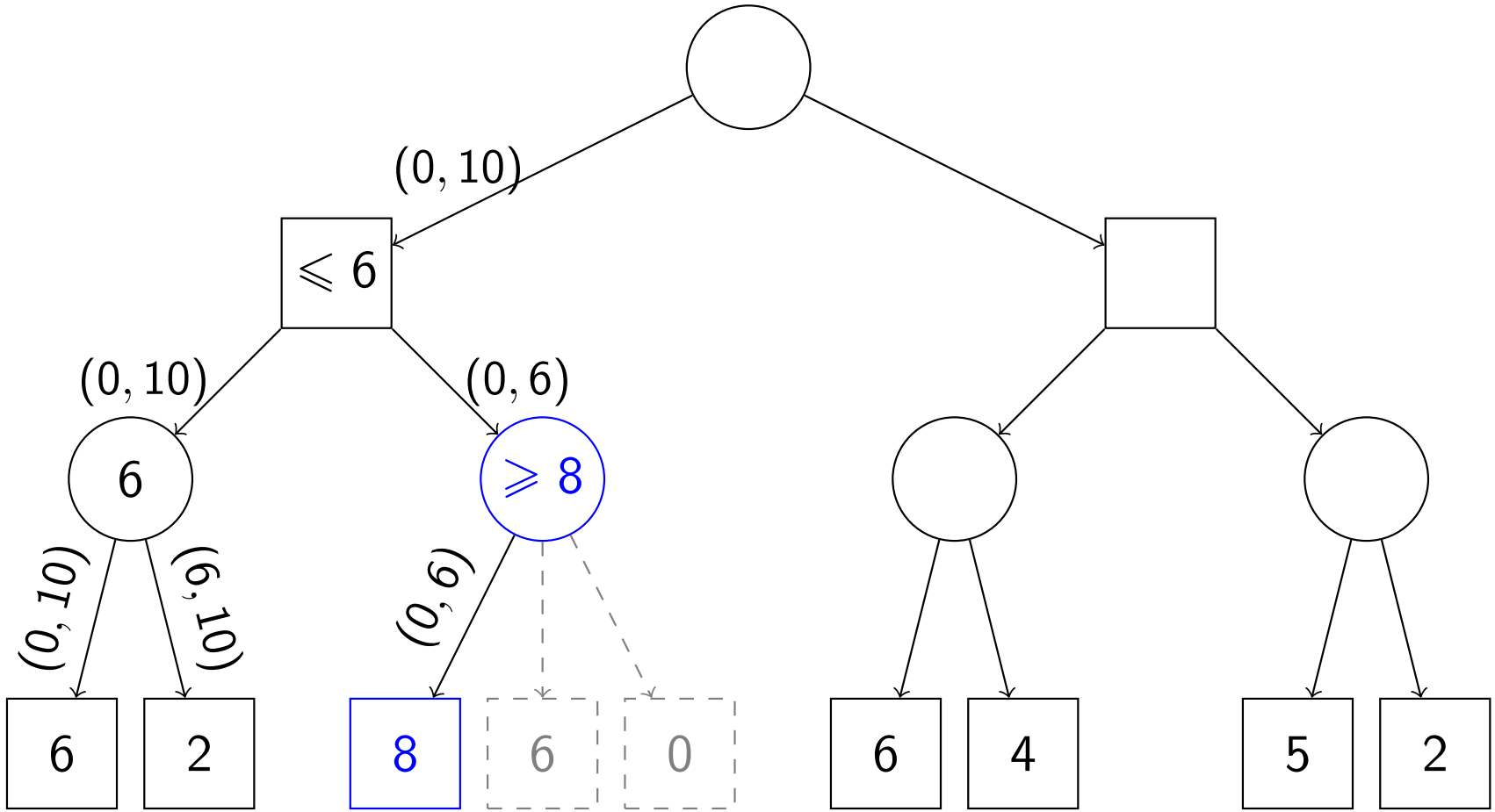
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



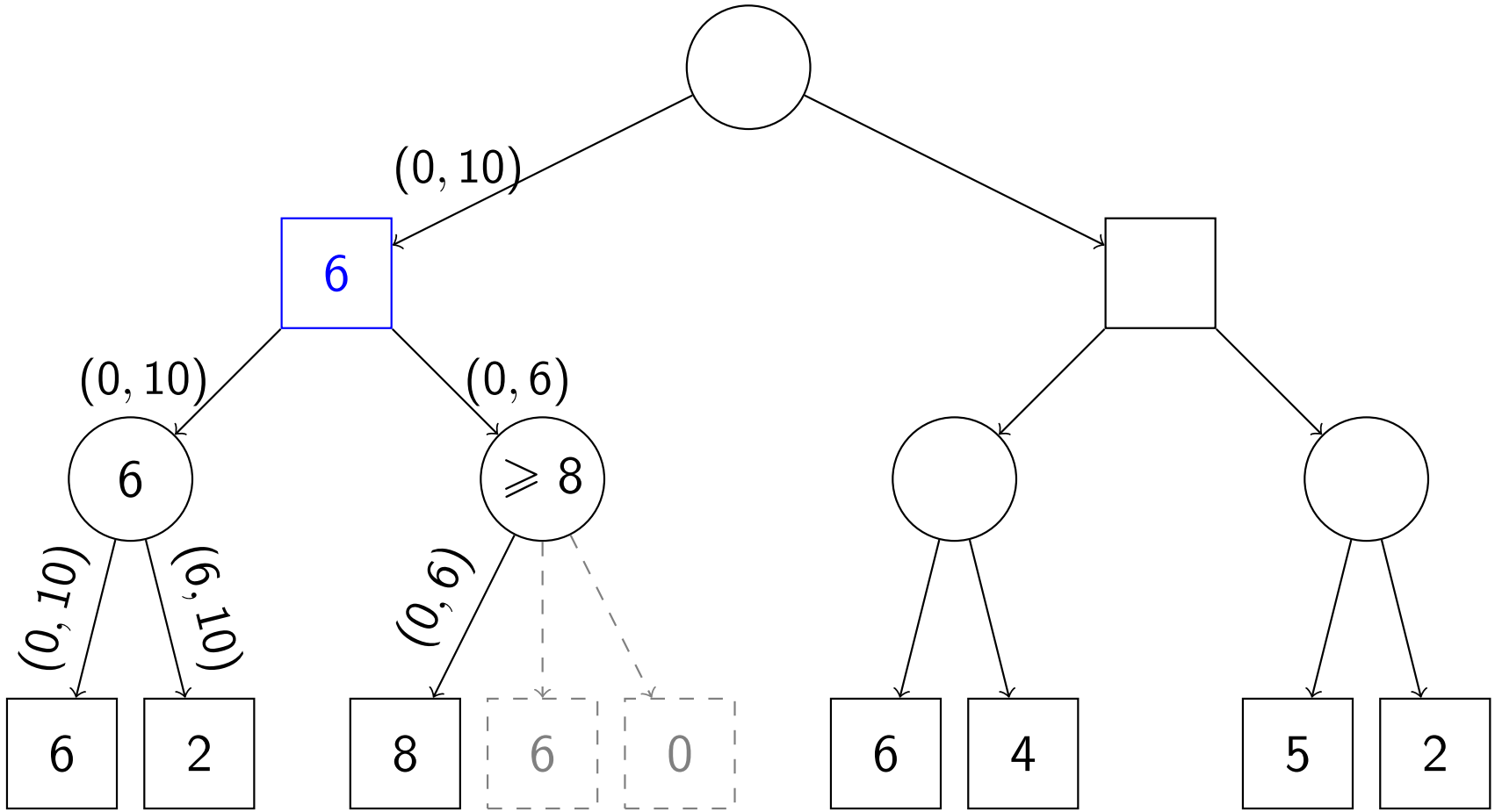
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



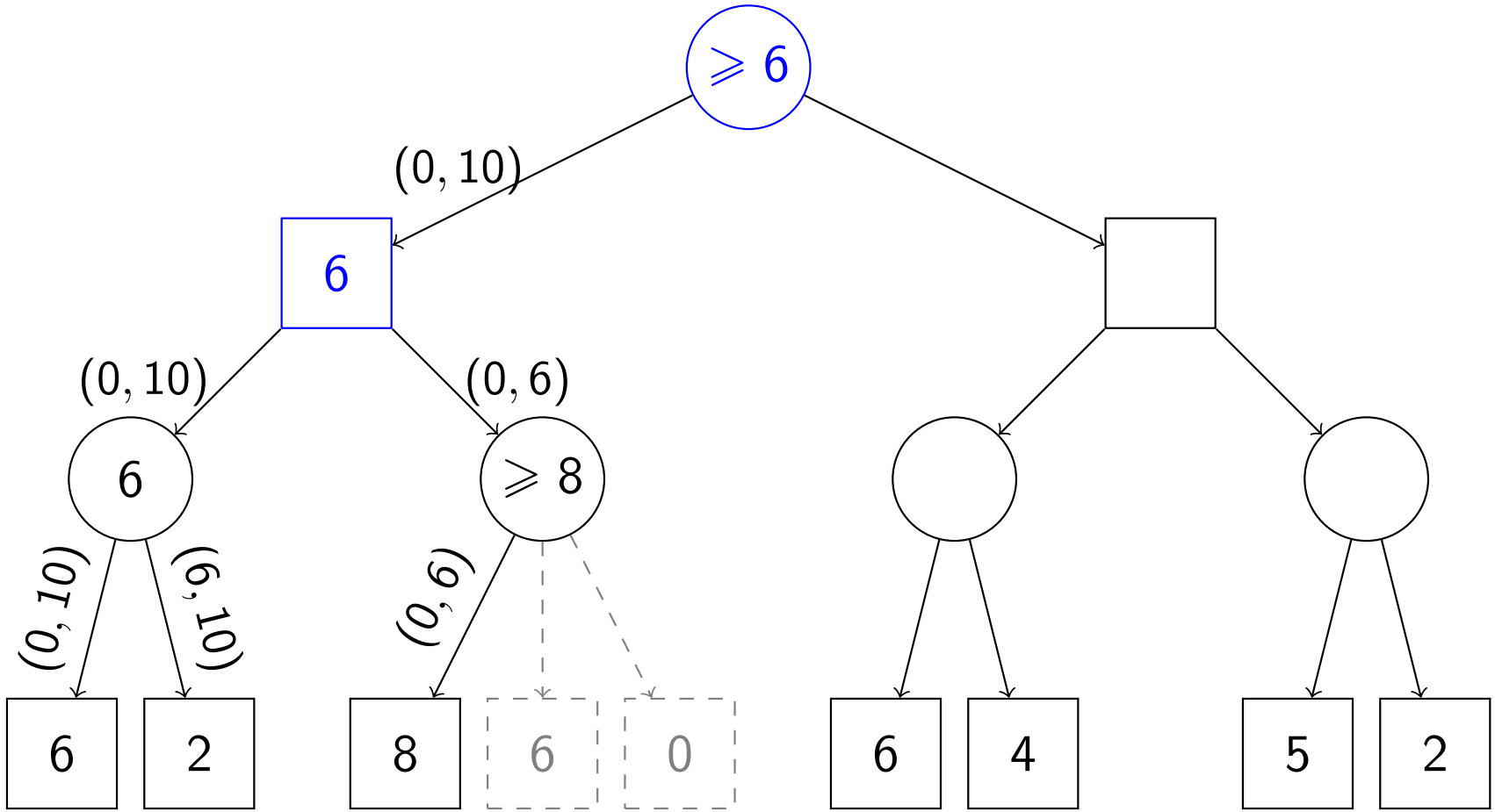
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



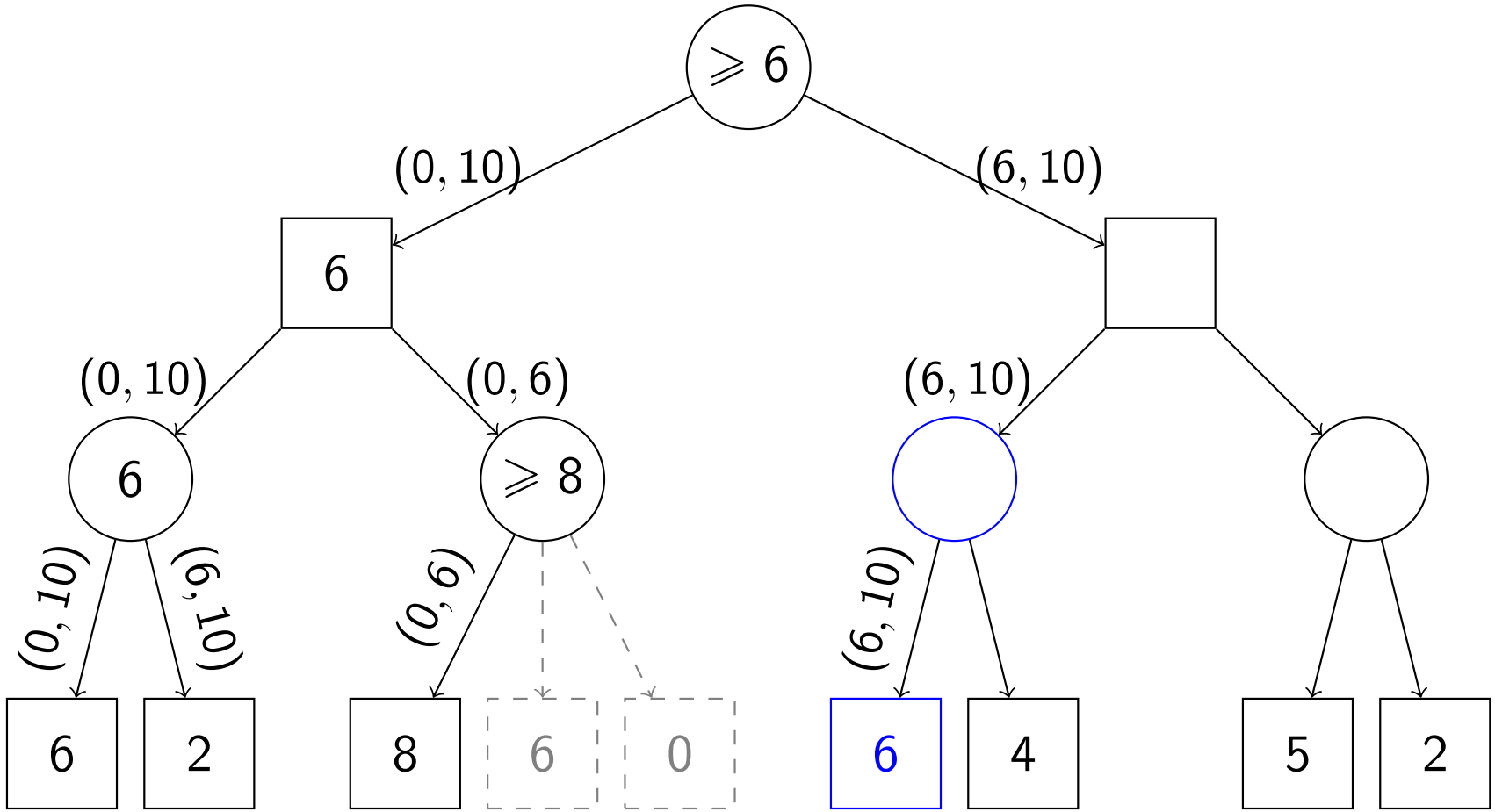
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



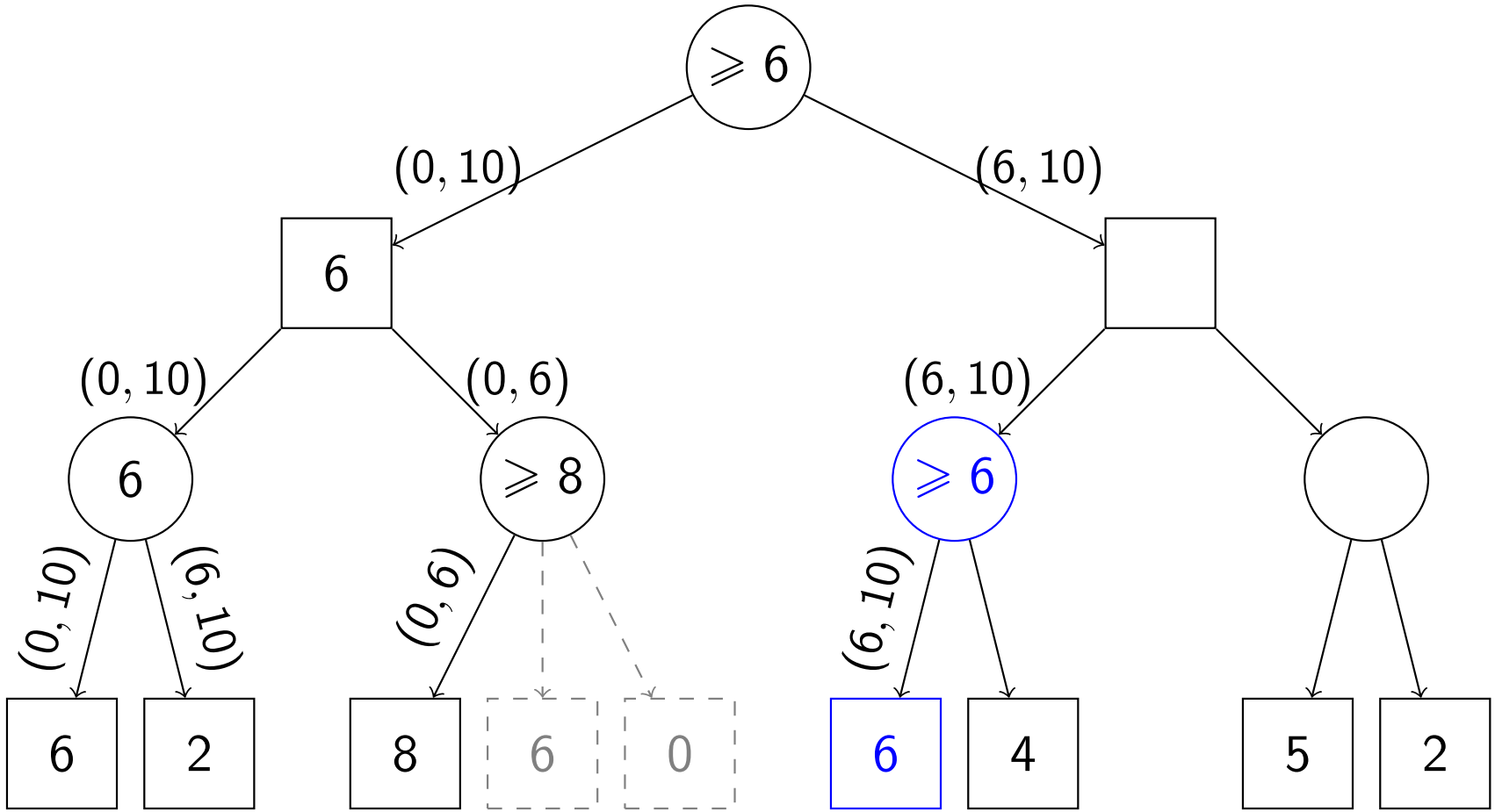
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



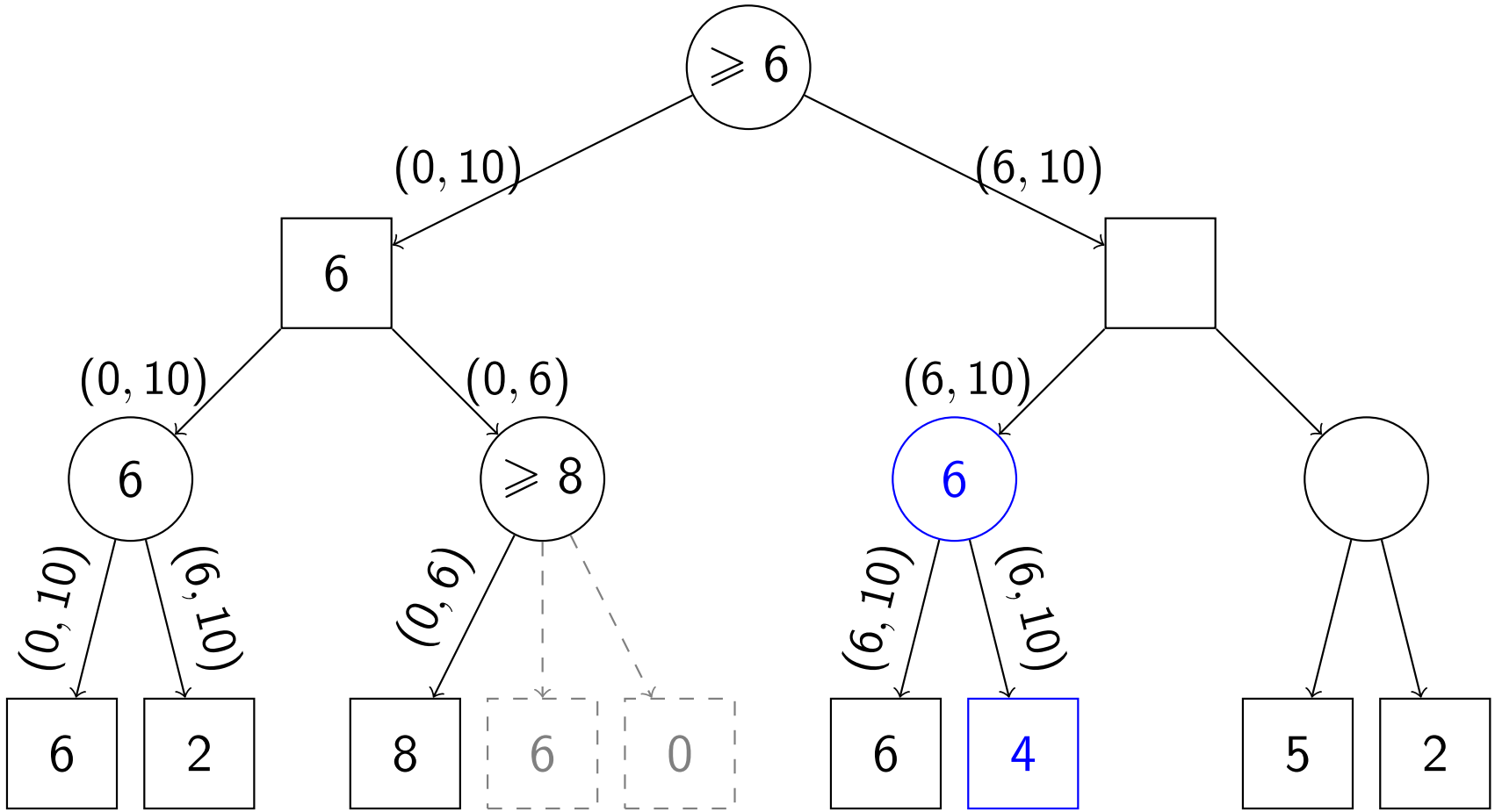
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



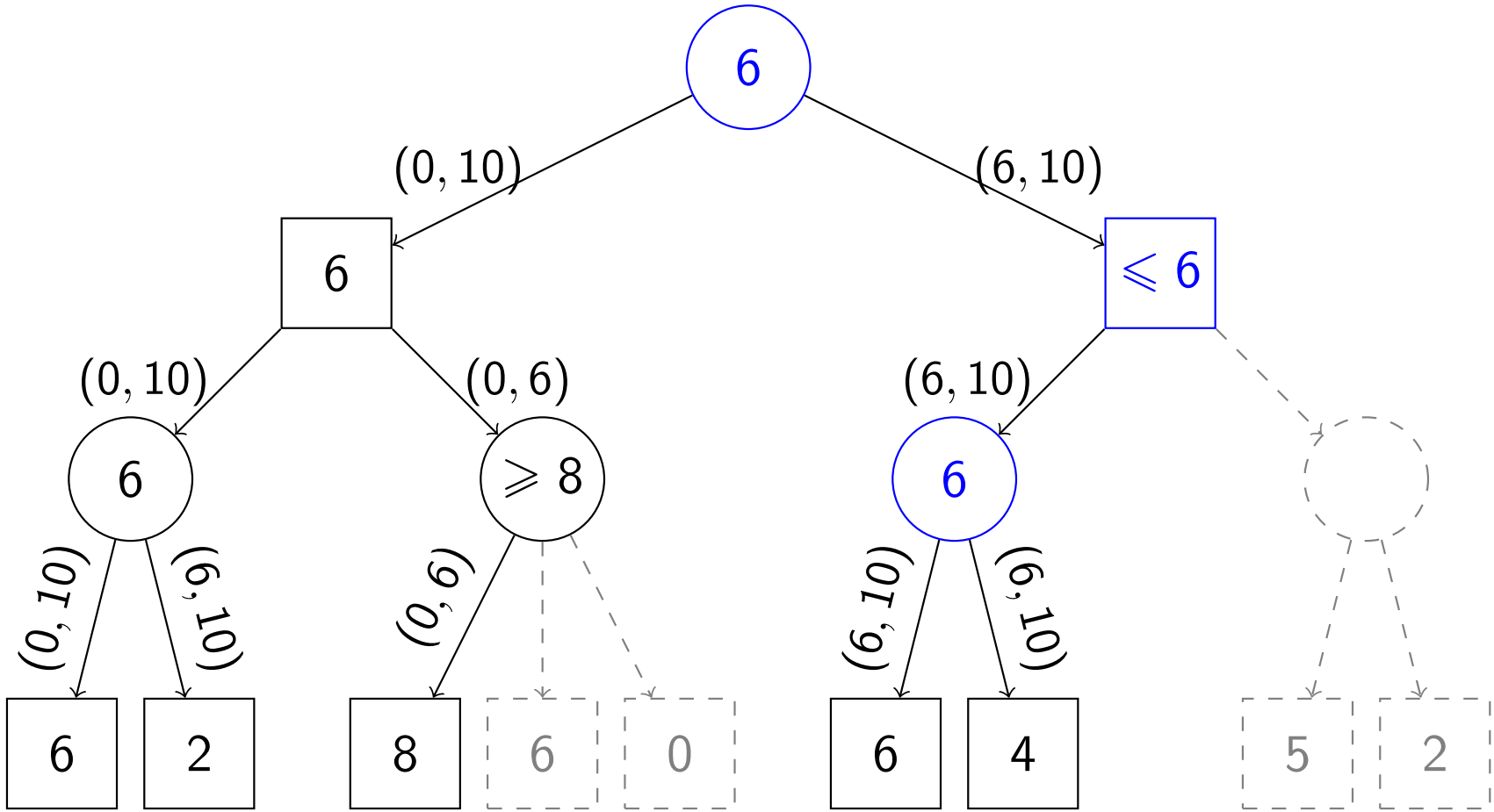
○ – ruch gracza *Max* □ – ruch gracza *Min*

Algorytm α-β [Knuth 1975] – przykład



○ – ruch gracza *Max* □ – ruch gracza *Min*

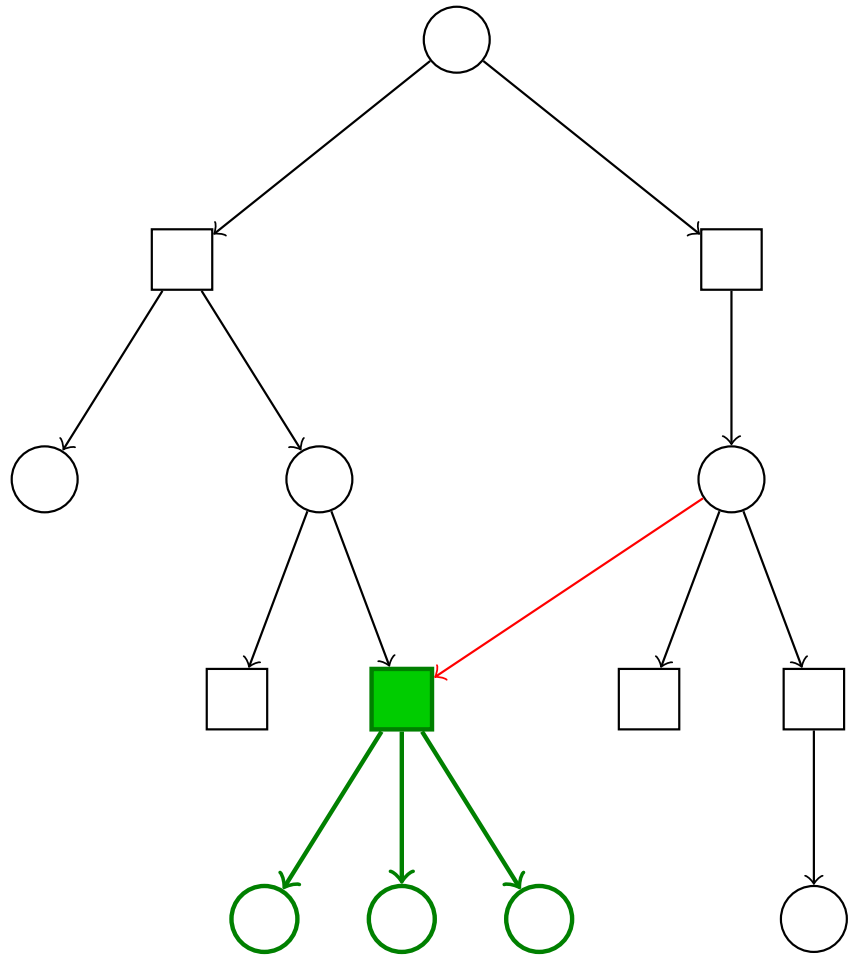
Algorytm α-β [Knuth 1975] – przykład



○ – ruch gracza *Max* □ – ruch gracza *Min*

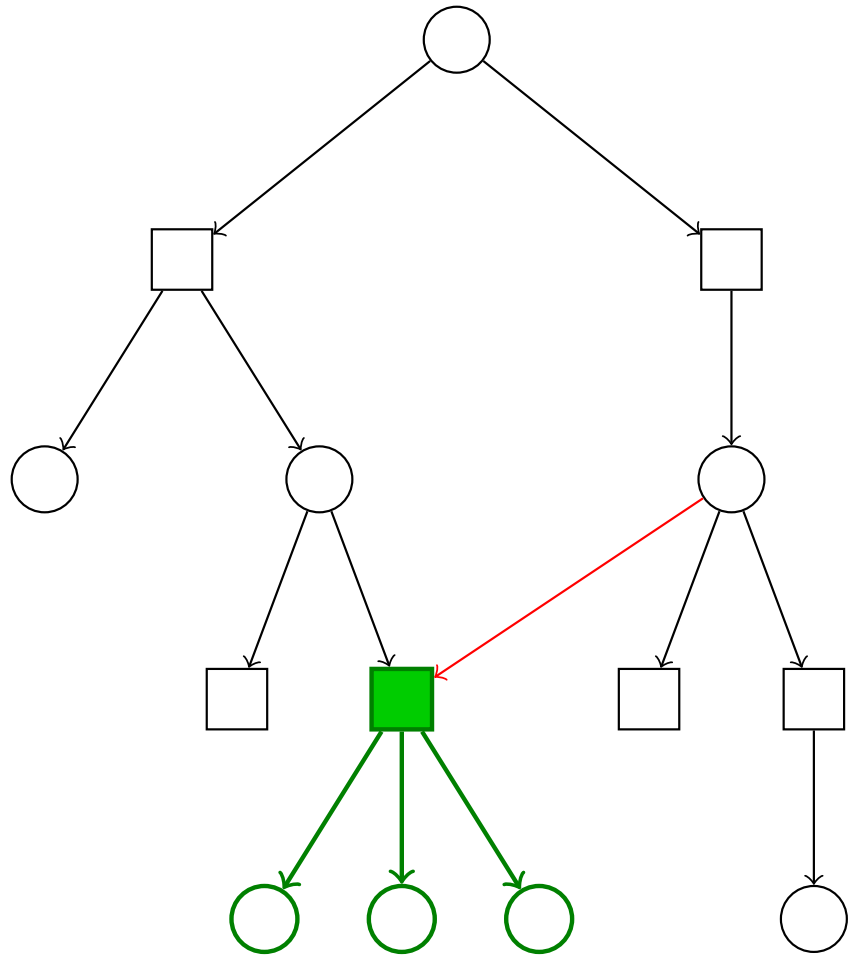
Tablica Transpozycji

- cel: uniknie redundantnych obliczeń;
- **transpozycja** – różne sekwencje ruchów prowadzą do tej samej pozycji;
- **tablica transpozycji** zawiera wyniki uzyskane dla dotychczas zbadanych pozycji.



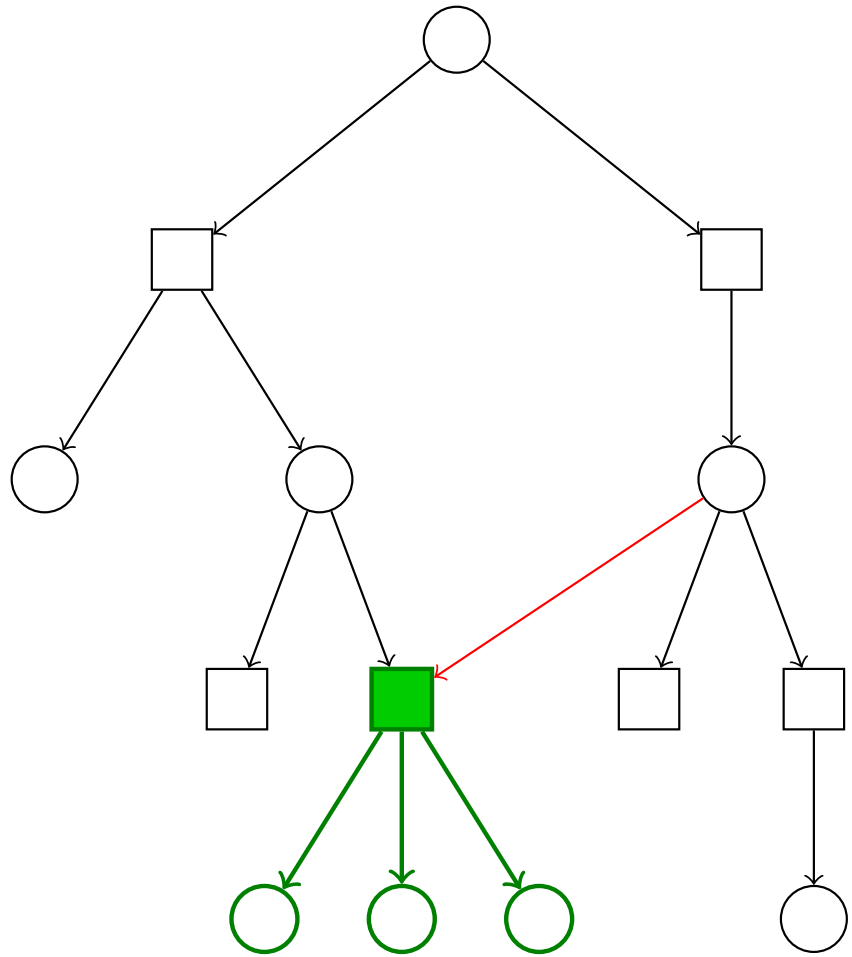
Tablica Transpozycji

- cel: uniknie redundantnych obliczeń;
- **transpozycja** – różne sekwencje ruchów prowadzą do tej samej pozycji;
- **tablica transpozycji** zawiera wyniki uzyskane dla dotychczas zbadanych pozycji.

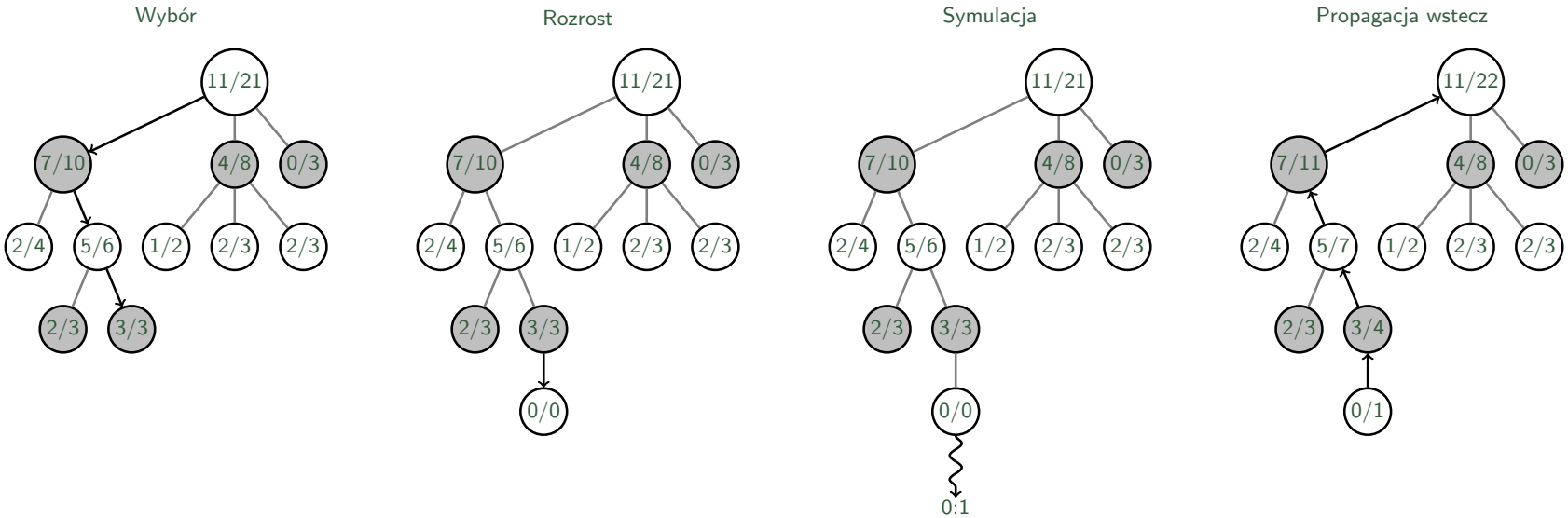


Tablica Transpozycji

- cel: uniknie redundantnych obliczeń;
- **transpozycja** – różne sekwencje ruchów prowadzą do tej samej pozycji;
- **tablica transpozycji** zawiera wyniki uzyskane dla dotychczas zbadanych pozycji.



Monte Carlo Tree Search [Kocsis 2006] – przegląd

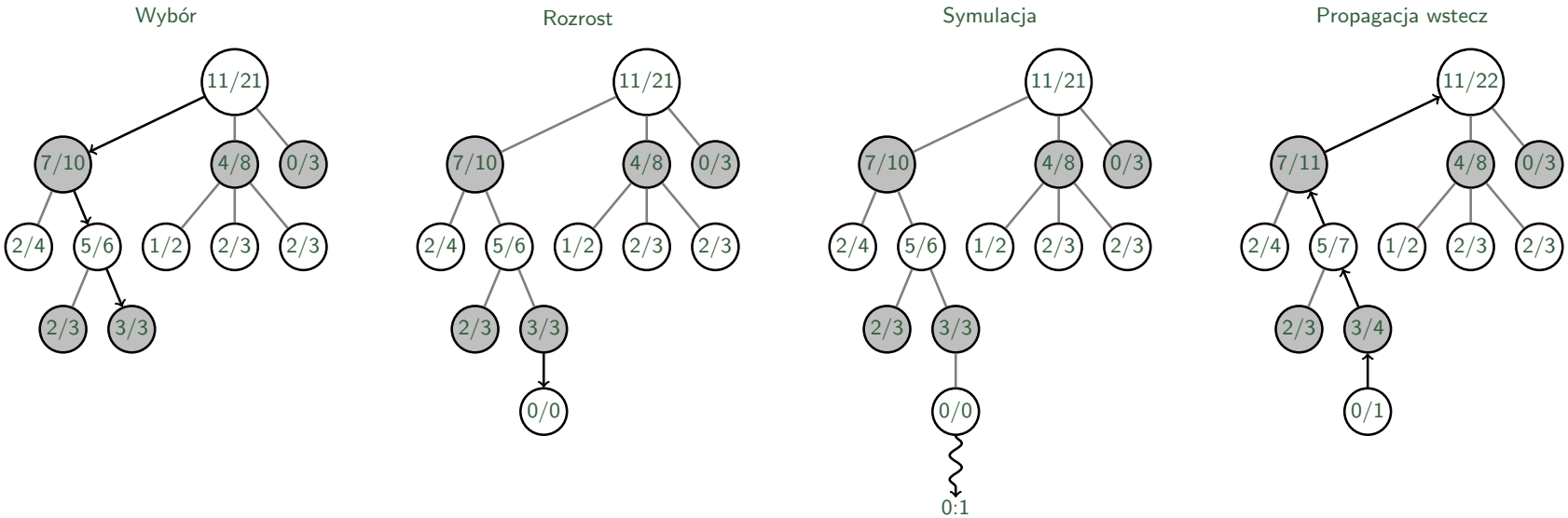


MCTS trzyma w pamięci i przyrostowo buduje drzewo poszukiwań, wykonując w pętli, dopóki jest czas:

1. wybór liścia L – preferencja najbardziej prawdopodobnych oraz najsłabiej zbadanych przebiegów gry;
2. rozrost – warunkowe dodanie do drzewa następników L;
3. symulacja – dokończenie rozgrywki (zazwyczaj losowe);
4. propagacja wstecz – na podstawie wyniku rozgrywki aktualizowane są informacje (wykorzystywane na etapie wyboru) o węzłach na ścieżce od korzenia do L.

(rysunek pochodzi z Wikipedii – `http://commons.wikimedia.org/wiki/File:MCTS_%28Polish%29.svg`)

Monte Carlo Tree Search [Kocsis 2006] – przegląd

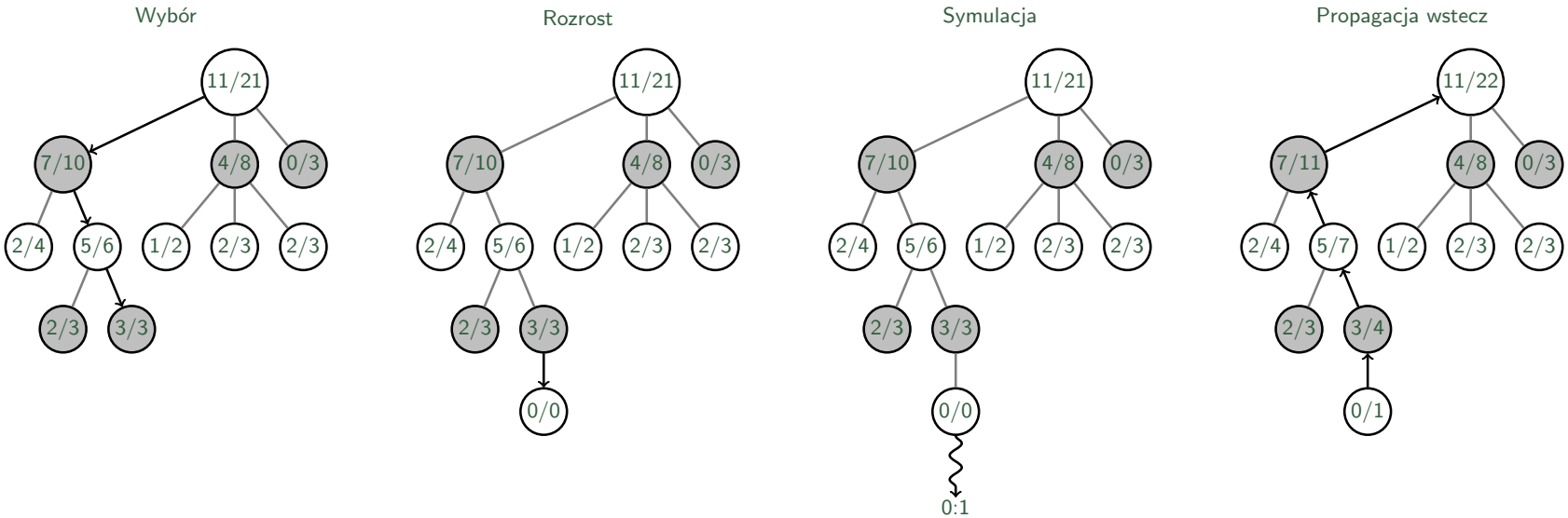


MCTS trzyma w pamięci i przyrostowo buduje drzewo poszukiwań, wykonując w pętli, dopóki jest czas:

1. wybór liścia L – preferencja najbardziej prawdopodobnych oraz najsłabiej zbadanych przebiegów gry;
2. rozrost – warunkowe dodanie do drzewa następników L;
3. symulacja – dokończenie rozgrywki (zazwyczaj losowe);
4. propagacja wstecz – na podstawie wyniku rozgrywki aktualizowane są informacje (wykorzystywane na etapie wyboru) o węzłach na ścieżce od korzenia do L.

(rysunek pochodzi z Wikipedii – `http://commons.wikimedia.org/wiki/File:MCTS_%28Polish%29.svg`)

Monte Carlo Tree Search [Kocsis 2006] – przegląd

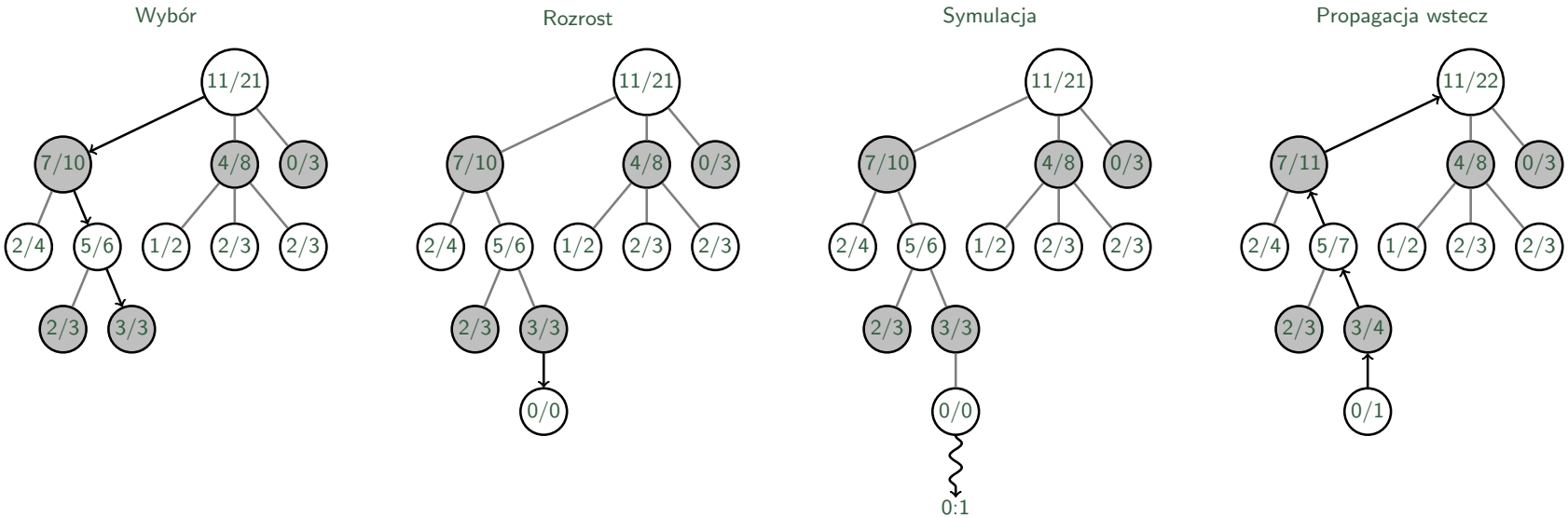


MCTS trzyma w pamięci i przyrostowo buduje drzewo poszukiwań, wykonując w pętli, dopóki jest czas:

1. wybór liścia L – preferencja najbardziej prawdopodobnych oraz najsłabiej zbadanych przebiegów gry;
2. rozrost – warunkowe dodanie do drzewa następników L;
3. symulacja – dokończenie rozgrywki (zazwyczaj losowe);
4. propagacja wstecz – na podstawie wyniku rozgrywki aktualizowane są informacje (wykorzystywane na etapie wyboru) o węzłach na ścieżce od korzenia do L.

(rysunek pochodzi z Wikipedii – `http://commons.wikimedia.org/wiki/File:MCTS_%28Polish%29.svg`)

Monte Carlo Tree Search [Kocsis 2006] – przegląd



MCTS trzyma w pamięci i przyrostowo buduje drzewo poszukiwań, wykonując w pętli, dopóki jest czas:

1. wybór liścia L – preferencja najbardziej prawdopodobnych oraz najsłabiej zbadanych przebiegów gry;
2. rozrost – warunkowe dodanie do drzewa następników L;
3. symulacja – dokończenie rozgrywki (zazwyczaj losowe);
4. propagacja wstecz – na podstawie wyniku rozgrywki aktualizowane są informacje (wykorzystywane na etapie wyboru) o węzłach na ścieżce od korzenia do L.

(rysunek pochodzi z Wikipedii – `http://commons.wikimedia.org/wiki/File:MCTS_%28Polish%29.svg`)

Monte Carlo Tree Search – drzewo

MCTS trzyma w pamięci i przyrostowo buduje drzewo (oznaczmy je T) będące spójnym fragmentem drzewa poszukiwań algorytmu Min-Max.

Początkowo T składa się jedynie z korzenia r, który reprezentuje pozycję dla której chcemy znaleźć suboptymalny ruch, oraz z jej synów.

Dla każdego węzła w w drzewie T pamiętane są dwie wartości:

- n_w – liczba dotychczasowych rozgrywek, które przebiegły przez w,
- v_w – suma wartości minimaksowych z tych rozgrywek (w przypadku gier typu wygrana-przegrana: liczba zwycięstw gracza Max w tych rozgrywkach).

Dla nowo dodanego do T węzła, obie te wartości inicjowane są liczbą 0.

Gdy $n_w > 0$, wartość minimaksową węzła w można oszacować ilorazem v_w/n_w. Pewność szacowania rośnie wraz z n_w.

Monte Carlo Tree Search – drzewo

MCTS trzyma w pamięci i przyrostowo buduje drzewo (oznaczmy je T) będące spójnym fragmentem drzewa poszukiwań algorytmu Min-Max.
Początkowo T składa się jedynie z korzenia r, który reprezentuje pozycję dla której chcemy znaleźć suboptymalny ruch, oraz z jej synów.

Dla każdego węzła w w drzewie T pamiętane są dwie wartości:

- n_w – liczba dotychczasowych rozgrywek, które przebiegły przez w,
- v_w – suma wartości minimaksowych z tych rozgrywek (w przypadku gier typu wygrana-przegrana: liczba zwycięstw gracza Max w tych rozgrywkach).

Dla nowo dodanego do T węzła, obie te wartości inicjowane są liczbą 0.
Gdy $n_w > 0$, wartość minimaksową węzła w można oszacować ilorazem v_w/n_w. Pewność szacowania rośnie wraz z n_w.

Monte Carlo Tree Search – drzewo

MCTS trzyma w pamięci i przyrostowo buduje drzewo (oznaczmy je T) będące spójnym fragmentem drzewa poszukiwań algorytmu Min-Max.
Początkowo T składa się jedynie z korzenia r, który reprezentuje pozycję dla której chcemy znaleźć suboptymalny ruch, oraz z jej synów.
Dla każdego węzła w w drzewie T pamiętane są dwie wartości:

- n_w – liczba dotychczasowych rozgrywek, które przebiegły przez w,
- v_w – suma wartości minimaksowych z tych rozgrywek (w przypadku gier typu wygrana-przegrana: liczba zwycięstw gracza Max w tych rozgrywkach).

Dla nowo dodanego do T węzła, obie te wartości inicjowane są liczbą 0.
Gdy $n_w > 0$, wartość minimaksową węzła w można oszacować ilorazem v_w/n_w. Pewność szacowania rośnie wraz z n_w.

Monte Carlo Tree Search – drzewo

MCTS trzyma w pamięci i przyrostowo buduje drzewo (oznaczmy je T) będące spójnym fragmentem drzewa poszukiwań algorytmu Min-Max.

Początkowo T składa się jedynie z korzenia r, który reprezentuje pozycję dla której chcemy znaleźć suboptymalny ruch, oraz z jej synów.

Dla każdego węzła w w drzewie T pamiętane są dwie wartości:

- n_w – liczba dotychczasowych rozgrywek, które przebiegły przez w,
- v_w – suma wartości minimaksowych z tych rozgrywek (w przypadku gier typu wygrana-przegrana: liczba zwycięstw gracza Max w tych rozgrywkach).

Dla nowo dodanego do T węzła, obie te wartości inicjowane są liczbą 0.

Gdy $n_w > 0$, wartość minimaksową węzła w można oszacować ilorazem v_w/n_w. Pewność szacowania rośnie wraz z n_w.

Monte Carlo Tree Search – drzewo

MCTS trzyma w pamięci i przyrostowo buduje drzewo (oznaczmy je T) będące spójnym fragmentem drzewa poszukiwań algorytmu Min-Max.
Początkowo T składa się jedynie z korzenia r, który reprezentuje pozycję dla której chcemy znaleźć suboptymalny ruch, oraz z jej synów.
Dla każdego węzła w w drzewie T pamiętane są dwie wartości:

- n_w – liczba dotychczasowych rozgrywek, które przebiegły przez w,
- v_w – suma wartości minimaksowych z tych rozgrywek (w przypadku gier typu wygrana-przegrana: liczba zwycięstw gracza Max w tych rozgrywkach).

Dla nowo dodanego do T węzła, obie te wartości inicjowane są liczbą 0.
Gdy $n_w > 0$, wartość minimaksową węzła w można oszacować ilorazem v_w/n_w. Pewność szacowania rośnie wraz z n_w.

Monte Carlo Tree Search – wybór liścia

Wybór jednego z liści drzewa T (oznaczmy tego liścia literą l).
Algorytm schodzi w dół drzewa T startując z korzenia r
i wybierając spośród następników każdego węzła w, taki węzeł x, który:

- albo nie był dotychczas wybrany (tj. spełniający $n_x = 0$),
- albo, gdy wszystkie następniki w były wcześniej co najmniej raz wybrane, optymalizuje pewną funkcję f (maksymalizuje albo minimalizuje ją w zależności od tego czy ruch w w należy do gracza Max czy Min).

Monte Carlo Tree Search – wybór liścia

Wybór jednego z liści drzewa T (oznaczmy tego liścia literą l). Algorytm schodzi w dół drzewa T startując z korzenia r i wybierając spośród następników każdego węzła w, taki węzeł x, który:

- albo nie był dotychczas wybrany (tj. spełniający $n_x = 0$),
- albo, gdy wszystkie następniki w były wcześniej co najmniej raz wybrane, optymalizuje pewną funkcję f (maksymalizuje albo minimalizuje ją w zależności od tego czy ruch w w należy do gracza Max czy Min).

Monte Carlo Tree Search – wybór liścia

Wybór jednego z liści drzewa T (oznaczmy tego liścia literą l).
Algorytm schodzi w dół drzewa T startując z korzenia r
i wybierając spośród następników każdego węzła w, taki węzeł x,
który:

- albo nie był dotychczas wybrany (tj. spełniający $n_x = 0$),
- albo, gdy wszystkie następniki w były wcześniej co najmniej raz wybrane, optymalizuje pewną funkcję f (maksymalizuje albo minimalizuje ją w zależności od tego czy ruch w w należy do gracza Max czy Min).

Monte Carlo Tree Search – wybór liścia

Wybór jednego z liści drzewa T (oznaczmy tego liścia literą l).
Algorytm schodzi w dół drzewa T startując z korzenia r
i wybierając spośród następników każdego węzła w, taki węzeł x,
który:

- albo nie był dotychczas wybrany (tj. spełniający $n_x = 0$),
- albo, gdy wszystkie następniki w były wcześniej co najmniej raz wybrane, optymalizuje pewną funkcję f (maksymalizuje albo minimalizuje ją w zależności od tego czy ruch w w należy do gracza Max czy Min).

Monte Carlo Tree Search – UCT

Funkcja f jest tak skonstruowana, że wartość $f(x)$ (dla x spełniającego $n_x > 0$):

- rośnie wraz z szacowaną wartością minimaksową x (tj. wraz z ilorazem v_x/n_x) – cel: częstsze wybieranie i dokładniejsze zbadanie bardziej prawdopodobnych przebiegów gry.
- równocześnie f powinna faworyzować także węzły słabo zbadane (o stosunkowo niskiej wartości n_x), dla których wspomniane szacowanie jest niepewne.

Najpopularniejsza odmiana MCTS zwana UCT (Upper Confidence Bounds for Trees), używa funkcji f w postaci zaproponowanej w [Kocsis 2006], znanej pod nazwą UCB (Upper Confidence Bounds):

$$f(x) = \begin{cases} \frac{v_x}{n_x} + c\sqrt{\frac{\ln n_w}{n_x}} & \text{gdy w } w \text{ idzie Max,} \\ \frac{v_x}{n_x} - c\sqrt{\frac{\ln n_w}{n_x}} & \text{gdy w } w \text{ idzie Min,} \end{cases}$$

gdzie $c > 0$ to stała, najczęściej dobierana eksperymentalnie (jedna z proponowanych wartość wynosi $\sqrt{2}$), zaś w jest ojcem węzła x w drzewie T.

Monte Carlo Tree Search – UCT

Funkcja f jest tak skonstruowana, że wartość $f(x)$ (dla x spełniającego $n_x > 0$):

- rośnie wraz z szacowaną wartością minimaksową x (tj. wraz z ilorazem v_x/n_x) – cel: częstsze wybieranie i dokładniejsze zbadanie bardziej prawdopodobnych przebiegów gry.
- równocześnie f powinna faworyzować także węzły słabo zbadane (o stosunkowo niskiej wartości n_x), dla których wspomniane szacowanie jest niepewne.

Najpopularniejsza odmiana MCTS zwana UCT (Upper Confidence Bounds for Trees), używa funkcji f w postaci zaproponowanej w [Kocsis 2006], znanej pod nazwą UCB (Upper Confidence Bounds):

$$f(x) = \begin{cases} \frac{v_x}{n_x} + c\sqrt{\frac{\ln n_w}{n_x}} & \text{gdy w } w \text{ idzie Max,} \\ \frac{v_x}{n_x} - c\sqrt{\frac{\ln n_w}{n_x}} & \text{gdy w } w \text{ idzie Min,} \end{cases}$$

gdzie $c > 0$ to stała, najczęściej dobierana eksperymentalnie (jedna z proponowanych wartość wynosi $\sqrt{2}$), zaś w jest ojcem węzła x w drzewie T.

Monte Carlo Tree Search – UCT

Funkcja f jest tak skonstruowana, że wartość $f(x)$ (dla x spełniającego $n_x > 0$):

- rośnie wraz z szacowaną wartością minimaksową x (tj. wraz z ilorazem v_x/n_x) – cel: częstsze wybieranie i dokładniejsze zbadanie bardziej prawdopodobnych przebiegów gry.
- równocześnie f powinna faworyzować także węzły słabo zbadane (o stosunkowo niskiej wartości n_x), dla których wspomniane szacowanie jest niepewne.

Najpopularniejsza odmiana MCTS zwana UCT (Upper Confidence Bounds for Trees), używa funkcji f w postaci zaproponowanej w [Kocsis 2006], znanej pod nazwą UCB (Upper Confidence Bounds):

$$f(x) = \begin{cases} \frac{v_x}{n_x} + c\sqrt{\frac{\ln n_w}{n_x}} & \text{gdy w } w \text{ idzie Max,} \\ \frac{v_x}{n_x} - c\sqrt{\frac{\ln n_w}{n_x}} & \text{gdy w } w \text{ idzie Min,} \end{cases}$$

gdzie $c > 0$ to stała, najczęściej dobierana eksperymentalnie (jedna z proponowanych wartość wynosi $\sqrt{2}$), zaś w jest ojcem węzła x w drzewie T.

Monte Carlo Tree Search – UCT

Funkcja f jest tak skonstruowana, że wartość $f(x)$ (dla x spełniającego $n_x > 0$):

- rośnie wraz z szacowaną wartością minimaksową x (tj. wraz z ilorazem v_x/n_x) – cel: częstsze wybieranie i dokładniejsze zbadanie bardziej prawdopodobnych przebiegów gry.
- równocześnie f powinna faworyzować także węzły słabo zbadane (o stosunkowo niskiej wartości n_x), dla których wspomniane szacowanie jest niepewne.

Najpopularniejsza odmiana MCTS zwana UCT (Upper Confidence Bounds for Trees), używa funkcji f w postaci zaproponowanej w [Kocsis 2006], znanej pod nazwą UCB (Upper Confidence Bounds):

$$f(x) = \begin{cases} \frac{v_x}{n_x} + c\sqrt{\frac{\ln n_w}{n_x}} & \text{gdy w } w \text{ idzie Max,} \\ \frac{v_x}{n_x} - c\sqrt{\frac{\ln n_w}{n_x}} & \text{gdy w } w \text{ idzie Min,} \end{cases}$$

gdzie $c > 0$ to stała, najczęściej dobierana eksperymentalnie (jedna z proponowanych wartość wynosi $\sqrt{2}$), zaś w jest ojcem węzła x w drzewie T.

Monte Carlo Tree Search – ekspansja

Ekspansja (rozrost):

- dodanie do drzewa T następników węzła l;
- może nastąpić tylko gdy l reprezentuje niekońcową pozycję gry;
- by zapobiec nadmiernemu rozrastaniu się drzewa T, zazwyczaj dokonuje się jej gdy zajdą dodatkowe warunki, np. gdy liczba n_l przekroczy pewien ustalony próg.

Jeśli ekspansja nastąpiła, to dalej przez l będziemy oznaczać liścia wybranego losowo spośród dodanych w jej wyniku do T.

Monte Carlo Tree Search – ekspansja

Ekspansja (rozrost):

- dodanie do drzewa T następników węzła l;
- może nastąpić tylko gdy l reprezentuje niekońcową pozycję gry;
- by zapobiec nadmiernemu rozrastaniu się drzewa T, zazwyczaj dokonuje się jej gdy zajdą dodatkowe warunki, np. gdy liczba n_l przekroczy pewien ustalony próg.

Jeśli ekspansja nastąpiła, to dalej przez l będziemy oznaczać liścia wybranego losowo spośród dodanych w jej wyniku do T.

Monte Carlo Tree Search – ekspansja

Ekspansja (rozrost):

- dodanie do drzewa T następników węzła l;
- może nastąpić tylko gdy l reprezentuje niekońcową pozycję gry;
- by zapobiec nadmiernemu rozrastaniu się drzewa T, zazwyczaj dokonuje się jej gdy zajdą dodatkowe warunki, np. gdy liczba n_l przekroczy pewien ustalony próg.

Jeśli ekspansja nastąpiła, to dalej przez l będziemy oznaczać liścia wybranego losowo spośród dodanych w jej wyniku do T.

Monte Carlo Tree Search – ekspansja

Ekspansja (rozrost):

- dodanie do drzewa T następników węzła l;
- może nastąpić tylko gdy l reprezentuje niekońcową pozycję gry;
- by zapobiec nadmiernemu rozrastaniu się drzewa T, zazwyczaj dokonuje się jej gdy zajdą dodatkowe warunki, np. gdy liczba n_l przekroczy pewien ustalony próg.

Jeśli ekspansja nastąpiła, to dalej przez l będziemy oznaczać liścia wybranego losowo spośród dodanych w jej wyniku do T.

Monte Carlo Tree Search – ekspansja

Ekspansja (rozrost):

- dodanie do drzewa T następników węzła l;
- może nastąpić tylko gdy l reprezentuje niekońcową pozycję gry;
- by zapobiec nadmiernemu rozrastaniu się drzewa T, zazwyczaj dokonuje się jej gdy zajdą dodatkowe warunki, np. gdy liczba n_l przekroczy pewien ustalony próg.

Jeśli ekspansja nastąpiła, to dalej przez l będziemy oznaczać liścia wybranego losowo spośród dodanych w jej wyniku do T.

Monte Carlo Tree Search – symulacja i propagacja

Dokończenie rozgrywki (symulacja)

- wykonanie kolejnych ruchów, zaczynając od pozycji l i kończąc w pewnej końcowej pozycji t;
- ruchy mogą być wybierane całkowicie losowo;
- rozkład prawdopodobieństwa można uzależnić od specyficznej dla rozpatrywanej gry heurystyki, aby częściej wybierać ruchy poprawne z punktu widzenia wykonujących je uczestników – wzrośnie realizm symulowanych rozgrywek i, co za tym idzie, siła całego algorytmu.

Propagacja wstecz

Aktualizacja wartości n_w i v_w dla każdego węzła w na ścieżce od korzenia r do wybranego liścia l, zwiększenie:

- n_w o 1;
- v_w o wypłatę gracza Max po zakończeniu gry w pozycji t (zazwyczaj 1 gdy wygrał, 0 gdy przegrał).

Monte Carlo Tree Search – symulacja i propagacja

Dokończenie rozgrywki (symulacja)

- wykonanie kolejnych ruchów, zaczynając od pozycji l i kończąc w pewnej końcowej pozycji t;
- ruchy mogą być wybierane całkowicie losowo;
- rozkład prawdopodobieństwa można uzależnić od specyficznej dla rozpatrywanej gry heurystyki, aby częściej wybierać ruchy poprawne z punktu widzenia wykonujących je uczestników – wzrośnie realizm symulowanych rozgrywek i, co za tym idzie, siła całego algorytmu.

Propagacja wstecz

Aktualizacja wartości n_w i v_w dla każdego węzła w na ścieżce od korzenia r do wybranego liścia l, zwiększenie:

- n_w o 1;
- v_w o wypłatę gracza Max po zakończeniu gry w pozycji t (zazwyczaj 1 gdy wygrał, 0 gdy przegrał).

Monte Carlo Tree Search – symulacja i propagacja

Dokończenie rozgrywki (symulacja)

- wykonanie kolejnych ruchów, zaczynając od pozycji l i kończąc w pewnej końcowej pozycji t;
- ruchy mogą być wybierane całkowicie losowo;
- rozkład prawdopodobieństwa można uzależnić od specyficznej dla rozpatrywanej gry heurystyki, aby częściej wybierać ruchy poprawne z punktu widzenia wykonujących je uczestników – wzrośnie realizm symulowanych rozgrywek i, co za tym idzie, siła całego algorytmu.

Propagacja wstecz

Aktualizacja wartości n_w i v_w dla każdego węzła w na ścieżce od korzenia r do wybranego liścia l, zwiększenie:

- n_w o 1;
- v_w o wypłatę gracza Max po zakończeniu gry w pozycji t (zazwyczaj 1 gdy wygrał, 0 gdy przegrał).

Monte Carlo Tree Search – symulacja i propagacja

Dokończenie rozgrywki (symulacja)

- wykonanie kolejnych ruchów, zaczynając od pozycji l i kończąc w pewnej końcowej pozycji t;
- ruchy mogą być wybierane całkowicie losowo;
- rozkład prawdopodobieństwa można uzależnić od specyficznej dla rozpatrywanej gry heurystyki, aby częściej wybierać ruchy poprawne z punktu widzenia wykonujących je uczestników – wzrośnie realizm symulowanych rozgrywek i, co za tym idzie, siła całego algorytmu.

Propagacja wstecz

Aktualizacja wartości n_w i v_w dla każdego węzła w na ścieżce od korzenia r do wybranego liścia l, zwiększenie:

- n_w o 1;
- v_w o wypłatę gracza Max po zakończeniu gry w pozycji t (zazwyczaj 1 gdy wygrał, 0 gdy przegrał).

Monte Carlo Tree Search – symulacja i propagacja

Dokończenie rozgrywki (symulacja)

- wykonanie kolejnych ruchów, zaczynając od pozycji l i kończąc w pewnej końcowej pozycji t;
- ruchy mogą być wybierane całkowicie losowo;
- rozkład prawdopodobieństwa można uzależnić od specyficznej dla rozpatrywanej gry heurystyki, aby częściej wybierać ruchy poprawne z punktu widzenia wykonujących je uczestników – wzrośnie realizm symulowanych rozgrywek i, co za tym idzie, siła całego algorytmu.

Propagacja wstecz

Aktualizacja wartości n_w i v_w dla każdego węzła w na ścieżce od korzenia r do wybranego liścia l, zwiększenie:

- n_w o 1;
- v_w o wypłatę gracza Max po zakończeniu gry w pozycji t (zazwyczaj 1 gdy wygrał, 0 gdy przegrał).

Monte Carlo Tree Search – symulacja i propagacja

Dokończenie rozgrywki (symulacja)

- wykonanie kolejnych ruchów, zaczynając od pozycji l i kończąc w pewnej końcowej pozycji t;
- ruchy mogą być wybierane całkowicie losowo;
- rozkład prawdopodobieństwa można uzależnić od specyficznej dla rozpatrywanej gry heurystyki, aby częściej wybierać ruchy poprawne z punktu widzenia wykonujących je uczestników – wzrośnie realizm symulowanych rozgrywek i, co za tym idzie, siła całego algorytmu.

Propagacja wstecz

Aktualizacja wartości n_w i v_w dla każdego węzła w na ścieżce od korzenia r do wybranego liścia l, zwiększenie:

- n_w o 1;
- v_w o wypłatę gracza Max po zakończeniu gry w pozycji t (zazwyczaj 1 gdy wygrał, 0 gdy przegrał).

Monte Carlo Tree Search – wybór ruchu

Gdy skończy się czas na podjęcie decyzji, algorytm, spośród następników węzła r, wybiera taki ruch m, który maksymalizuje wartość:

- v_m/n_m
  (najwyżej oceniany ruch);
- albo, alternatywnie, n_m
  (najczęściej odwiedzany ruch $\approxeq$ najwyżej oceniany przez większość czasu pracy algorytmu);
- rzadziej stosuje się też inne kryterium.

Monte Carlo Tree Search – wybór ruchu

Gdy skończy się czas na podjęcie decyzji, algorytm, spośród następników węzła r, wybiera taki ruch m, który maksymalizuje wartość:

- v_m/n_m
  (najwyżej oceniany ruch);
- albo, alternatywnie, n_m
  (najczęściej odwiedzany ruch $\approxeq$ najwyżej oceniany przez większość czasu pracy algorytmu);
- rzadziej stosuje się też inne kryterium.

Monte Carlo Tree Search – wybór ruchu

Gdy skończy się czas na podjęcie decyzji, algorytm, spośród następników węzła r, wybiera taki ruch m, który maksymalizuje wartość:

- v_m/n_m
  (najwyżej oceniany ruch);
- albo, alternatywnie, n_m
  (najczęściej odwiedzany ruch $\approxeq$ najwyżej oceniany przez większość czasu pracy algorytmu);
- rzadziej stosuje się też inne kryterium.

Monte Carlo Tree Search – wybór ruchu

Gdy skończy się czas na podjęcie decyzji, algorytm, spośród następników węzła r, wybiera taki ruch m, który maksymalizuje wartość:

- v_m/n_m
  (najwyżej oceniany ruch);
- albo, alternatywnie, n_m
  (najczęściej odwiedzany ruch $\approxeq$ najwyżej oceniany przez większość czasu pracy algorytmu);
- rzadziej stosuje się też inne kryterium.

Monte Carlo Tree Search – główne zalety

- nie wymaga wiedzy dziedzinowej;
- jednocześnie pozwala taką wiedzę wykorzystać (na etapie selekcji oraz symulacji);
- możliwość podania aktualnego szacowania wyniku w dowolnym momencie, co umożliwia bardzo dokładne dostosowanie się do czasu dostępnego na podjęcie decyzji;
- możliwość dość łatwego zrównoleglenia obliczeń;

Przegląd różnych wariantów MCTS i ich zastosowań można znaleźć w [Browne 2012]

C. Browne, et al. *A Survey of Monte Carlo Tree Search Methods*, IEEE Transactions on Computational Intelligence and AI in Games, 2012, Volume 4, Issue 1

Monte Carlo Tree Search – główne zalety

- nie wymaga wiedzy dziedzinowej;
- jednocześnie pozwala taką wiedzę wykorzystać (na etapie selekcji oraz symulacji);
- możliwość podania aktualnego szacowania wyniku w dowolnym momencie, co umożliwia bardzo dokładne dostosowanie się do czasu dostępnego na podjęcie decyzji;
- możliwość dość łatwego zrównoleglenia obliczeń;

Przegląd różnych wariantów MCTS i ich zastosowań można znaleźć w [Browne 2012]

C. Browne, et al. *A Survey of Monte Carlo Tree Search Methods*, IEEE Transactions on Computational Intelligence and AI in Games, 2012, Volume 4, Issue 1

Monte Carlo Tree Search – główne zalety

- nie wymaga wiedzy dziedzinowej;
- jednocześnie pozwala taką wiedzę wykorzystać (na etapie selekcji oraz symulacji);
- możliwość podania aktualnego szacowania wyniku w dowolnym momencie, co umożliwia bardzo dokładne dostosowanie się do czasu dostępnego na podjęcie decyzji;
- możliwość dość łatwego zrównoleglenia obliczeń;

Przegląd różnych wariantów MCTS i ich zastosowań można znaleźć w [Browne 2012]

C. Browne, et al. *A Survey of Monte Carlo Tree Search Methods*, IEEE Transactions on Computational Intelligence and AI in Games, 2012, Volume 4, Issue 1

Monte Carlo Tree Search – główne zalety

- nie wymaga wiedzy dziedzinowej;
- jednocześnie pozwala taką wiedzę wykorzystać (na etapie selekcji oraz symulacji);
- możliwość podania aktualnego szacowania wyniku w dowolnym momencie, co umożliwia bardzo dokładne dostosowanie się do czasu dostępnego na podjęcie decyzji;
- możliwość dość łatwego zrównoleglenia obliczeń;

Przegląd różnych wariantów MCTS i ich zastosowań można znaleźć w [Browne 2012]

C. Browne, et al. *A Survey of Monte Carlo Tree Search Methods*, IEEE Transactions on Computational Intelligence and AI in Games, 2012, Volume 4, Issue 1

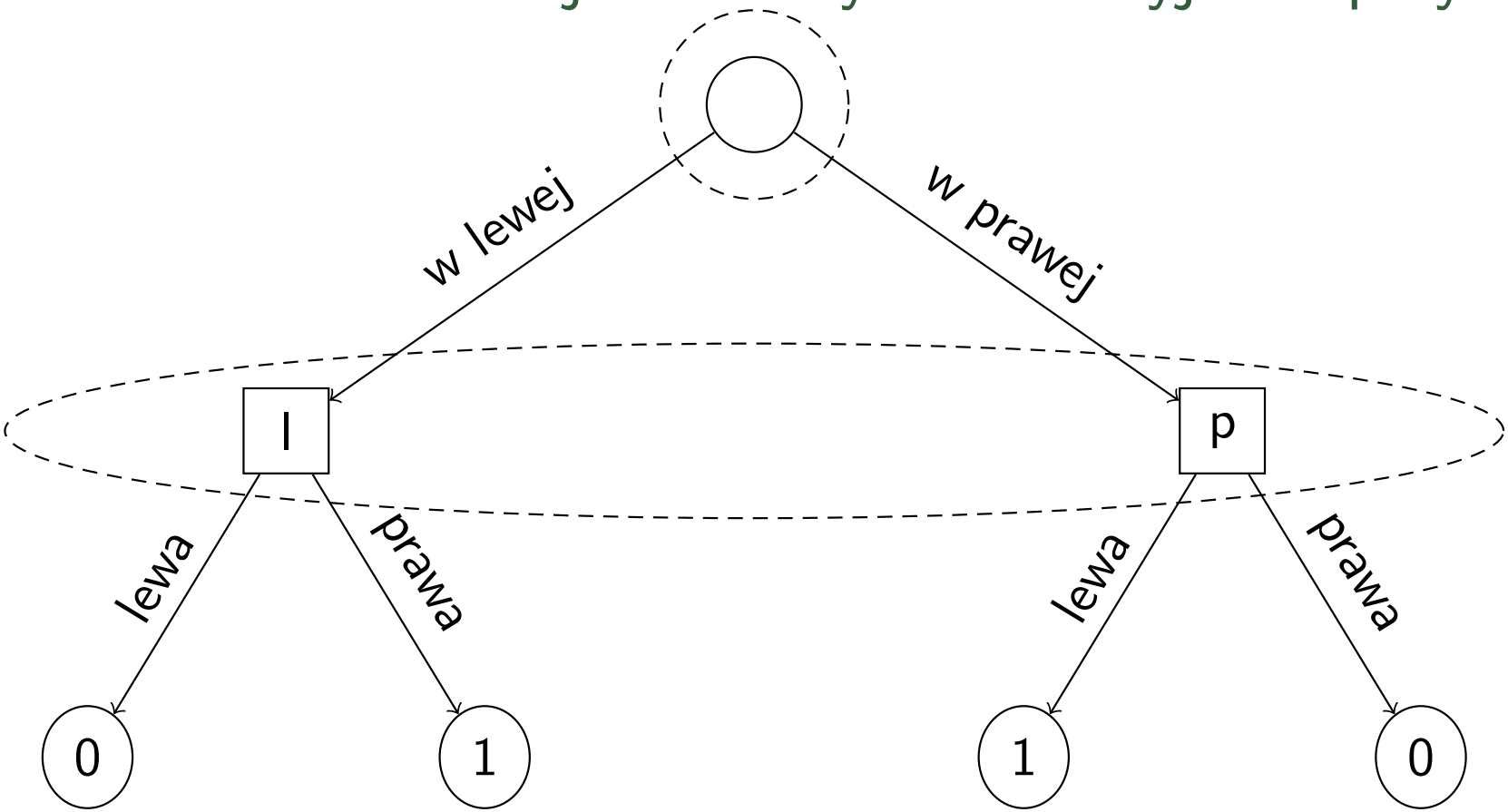
Monte Carlo Tree Search – główne zalety

- nie wymaga wiedzy dziedzinowej;
- jednocześnie pozwala taką wiedzę wykorzystać (na etapie selekcji oraz symulacji);
- możliwość podania aktualnego szacowania wyniku w dowolnym momencie, co umożliwia bardzo dokładne dostosowanie się do czasu dostępnego na podjęcie decyzji;
- możliwość dość łatwego zrównoleglenia obliczeń;

Przegląd różnych wariantów MCTS i ich zastosowań można znaleźć w [Browne 2012]

C. Browne, et al. *A Survey of Monte Carlo Tree Search Methods*, IEEE Transactions on Computational Intelligence and AI in Games, 2012, Volume 4, Issue 1

Niedoskonała informacja – zbiory informacyjne – przykład



Pierwszy gracz (Max) umieszcza przedmiot w lewej lub w prawej ręce, doprowadzając tym samym do pozycji *l* albo *p*. Drugi gracz (Min) wygra jeśli odgadnie w której ręce znajduje się przedmiot. Nie wie on jednak czy gra jest w pozycji *l* czy *p* (nie ma doskonałej informacji), co jest wyrażone poprzez umieszczenie obu pozycji we wspólnym zbiorze informacyjnym.

Determinizacja i PIMC

- **zbiór informacyjny** – zbiór pozycji w których może znajdować się w danej chwili gra,
- np. w grach karcianych w jednym zbiorze informacyjnym znajdą się pozycje o takim samym rozłożeniu kart widocznych dla gracza i różnym rozłożeniu kart niewidocznych;
- **determinizacja** – wybranie konkretnej pozycji ze zbioru informacyjnego (+ ewentualne ustalenie rezultatów wszelkich zdarzeń losowych);
  także otrzymana w ten sposób **gra z doskonałą informacją** (i deterministyczna);
- **Perfect Information Monte Carlo (PIMC)** – metoda podejmowania decyzji w grach bez doskonałej informacji, polegająca na niezależnym rozwiązaniu szeregu determinizacji i wybraniu zagrania dla którego otrzymano najlepszy średni wynik.

Determinizacja i PIMC

- **zbiór informacyjny** – zbiór pozycji w których może znajdować się w danej chwili gra,
- np. w grach karcianych w jednym zbiorze informacyjnym znajdą się pozycje o takim samym rozłożeniu kart widocznych dla gracza i różnym rozłożeniu kart niewidocznych;
- **determinizacja** – wybranie konkretnej pozycji ze zbioru informacyjnego (+ ewentualne ustalenie rezultatów wszelkich zdarzeń losowych);
  także otrzymana w ten sposób **gra z doskonałą informacją** (i deterministyczna);
- **Perfect Information Monte Carlo (PIMC)** – metoda podejmowania decyzji w grach bez doskonałej informacji, polegająca na niezależnym rozwiązaniu szeregu determinizacji i wybraniu zagrania dla którego otrzymano najlepszy średni wynik.

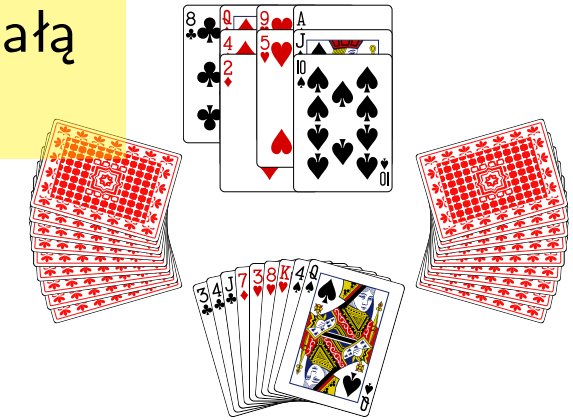
Determinizacja i PIMC

- **zbiór informacyjny** – zbiór pozycji w których może znajdować się w danej chwili gra,
- np. w grach karcianych w jednym zbiorze informacyjnym znajdą się pozycje o takim samym rozłożeniu kart widocznych dla gracza i różnym rozłożeniu kart niewidocznych;
- **determinizacja** – wybranie konkretnej pozycji ze zbioru informacyjnego (+ ewentualne ustalenie rezultatów wszelkich zdarzeń losowych);
  także otrzymana w ten sposób **gra z doskonałą informacją** (i deterministyczna);
- **Perfect Information Monte Carlo (PIMC)** – metoda podejmowania decyzji w grach bez doskonałej informacji, polegająca na niezależnym rozwiązaniu szeregu determinizacji i wybraniu zagrania dla którego otrzymano najlepszy średni wynik.

Determinizacja i PIMC

- **zbiór informacyjny** – zbiór pozycji w których może znajdować się w danej chwili gra,
- np. w grach karcianych w jednym zbiorze informacyjnym znajdą się pozycje o takim samym rozłożeniu kart widocznych dla gracza i różnym rozłożeniu kart niewidocznych;
- **determinizacja** – wybranie konkretnej pozycji ze zbioru informacyjnego (+ ewentualne ustalenie rezultatów wszelkich zdarzeń losowych);
  także otrzymana w ten sposób **gra z doskonałą informacją** (i deterministyczna);
- **Perfect Information Monte Carlo (PIMC)** – metoda podejmowania decyzji w grach bez doskonałej informacji, polegająca na niezależnym rozwiązaniu szeregu determinizacji i wybraniu zagrania dla którego otrzymano najlepszy średni wynik.

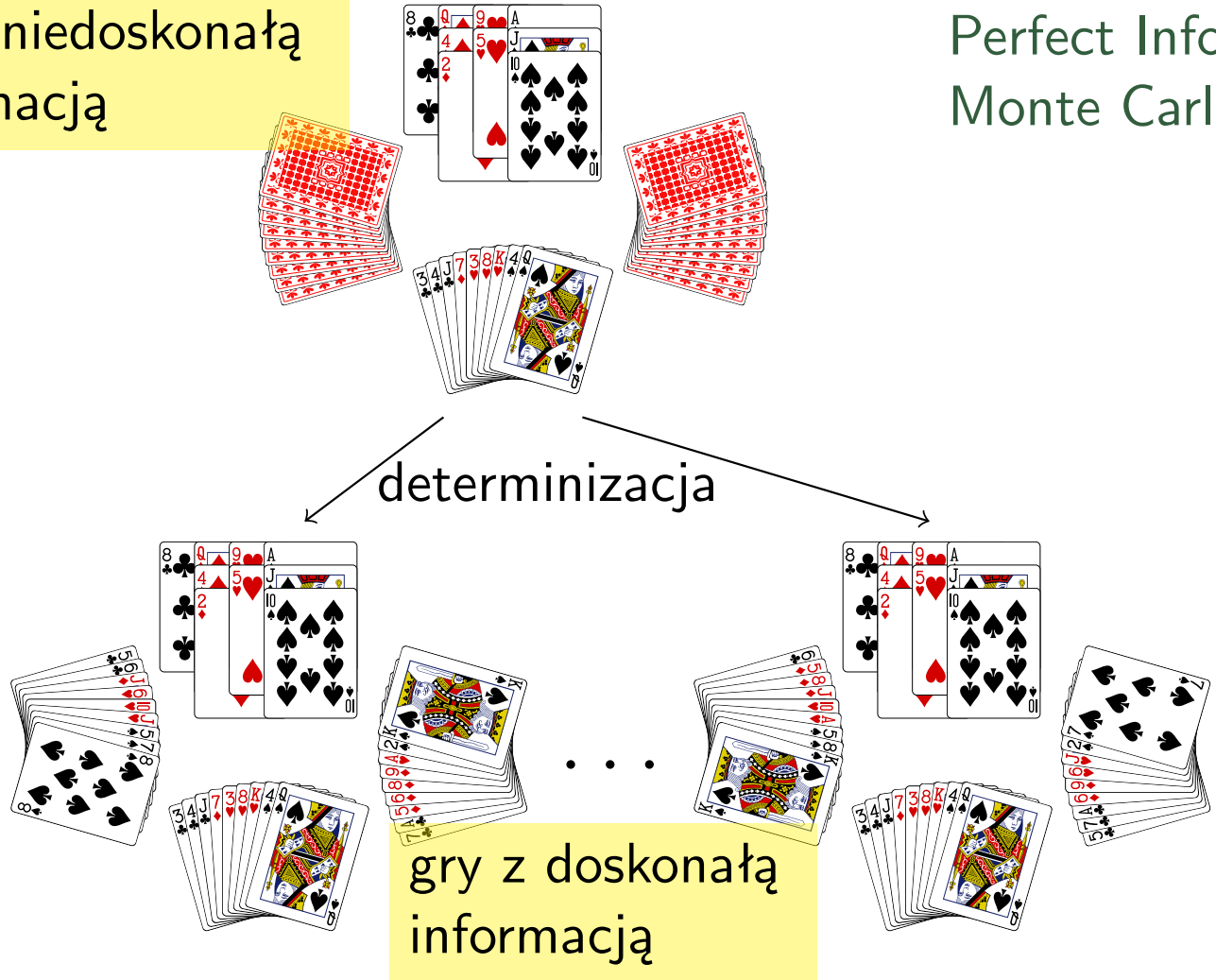
gra z niedoskonałą informacją



Perfect Information Monte Carlo

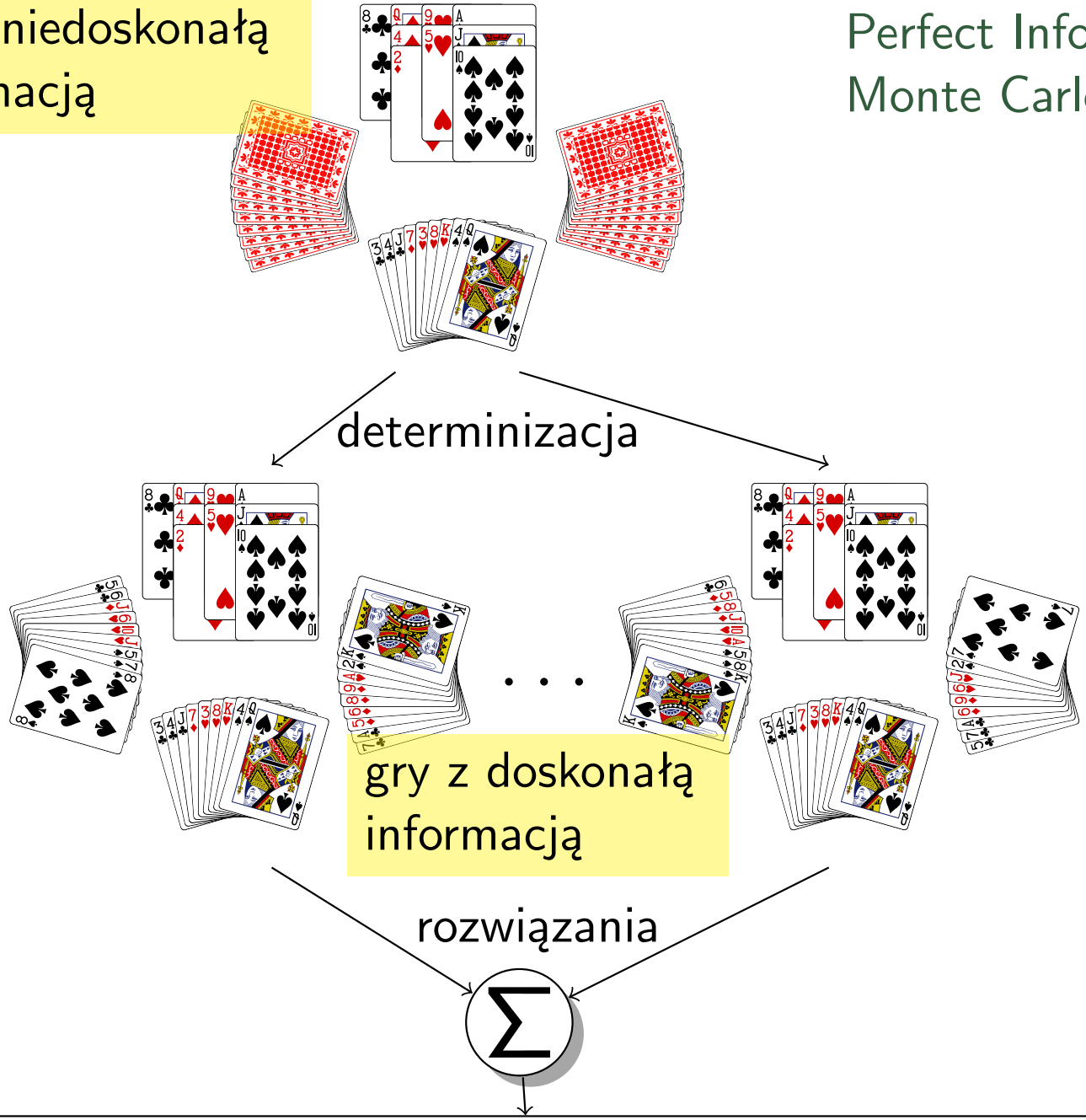
gra z niedoskonałą informacją

Perfect Information Monte Carlo



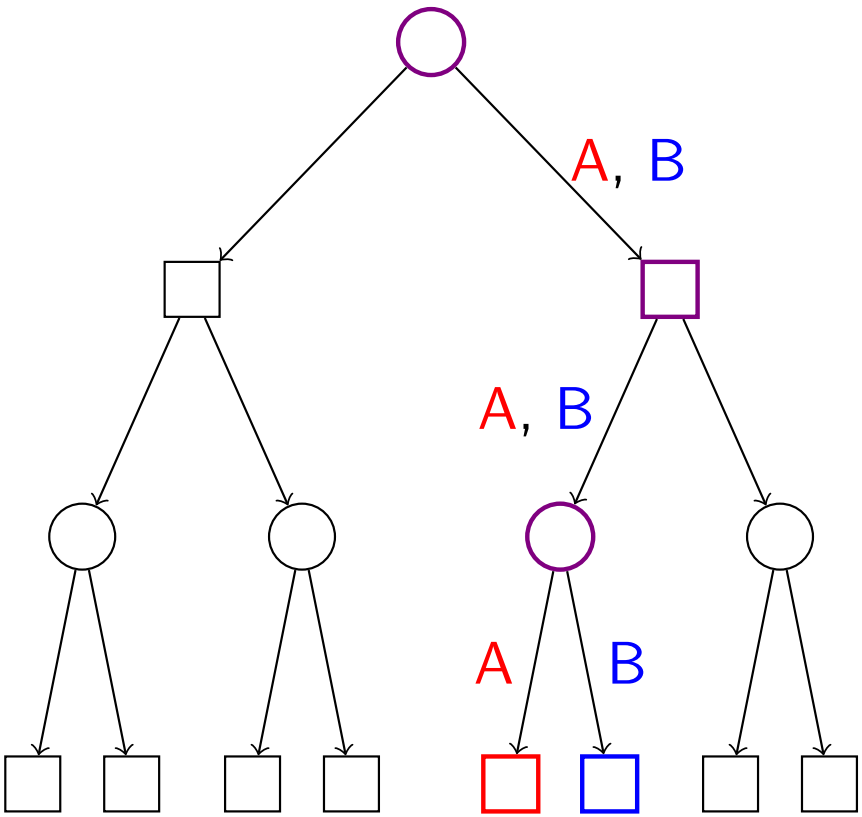
gra z niedoskonałą informacją

Perfect Information Monte Carlo



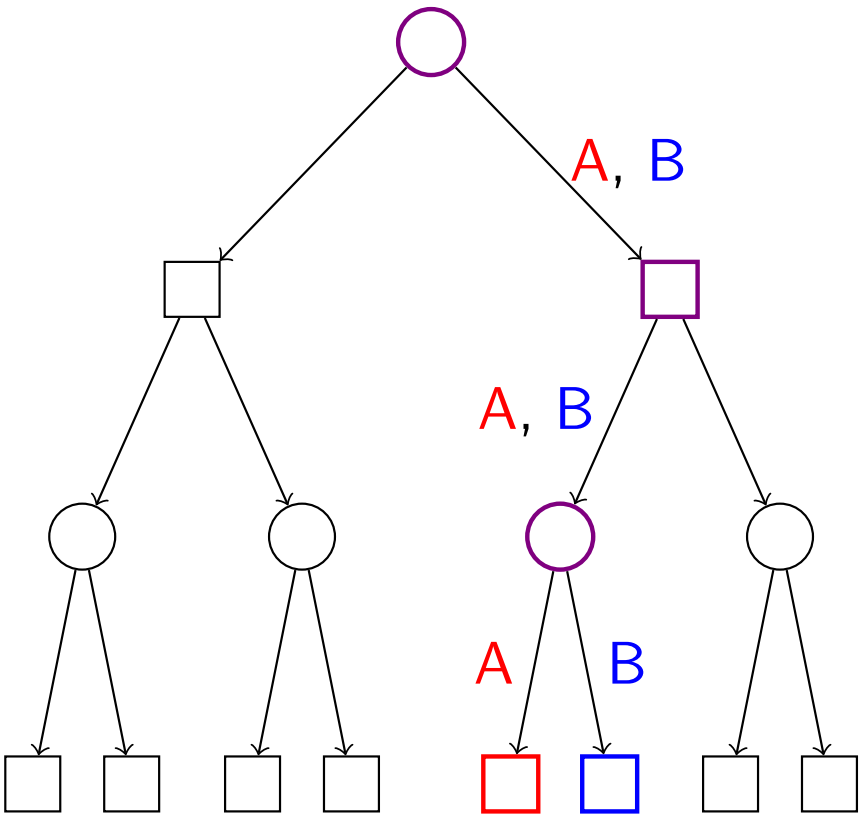
rozwiązanie oryginalnego problemu (z niedoskonałą informacją)

Problem łączenia strategii [Frank 1996]



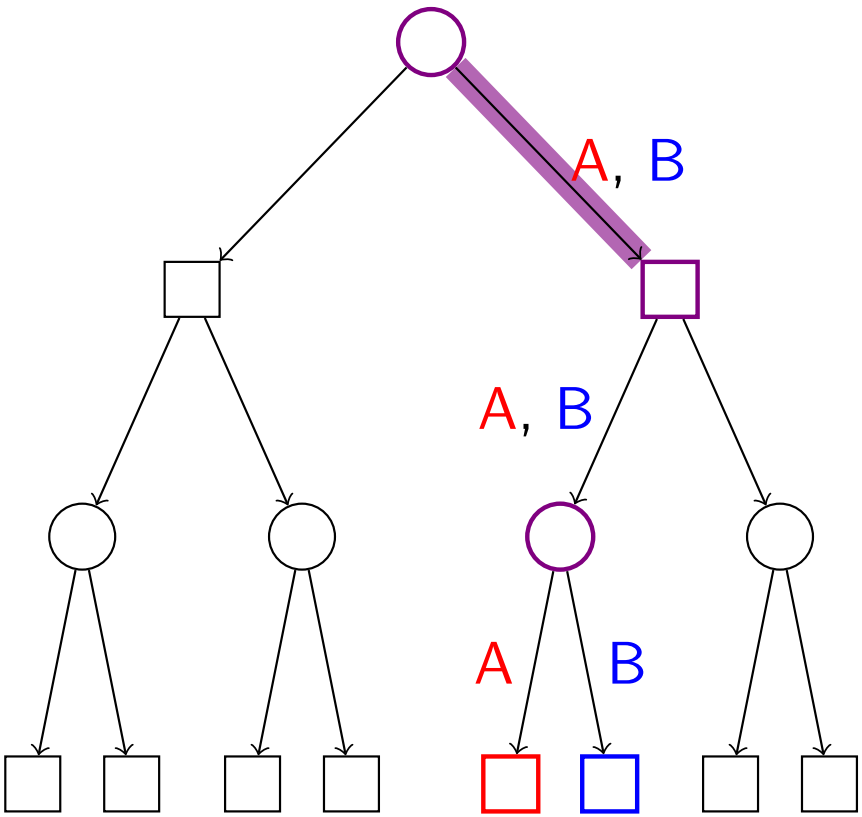
- ruch wygrywający w każdej z pozycji (determinizacji) **z osobna** nie gwarantuje wygranej w grze bez doskonałej informacji;
- znalezione dla poszczególnych determinizacji (np. A i B) zwycięskie strategie, pomimo wspólnego pierwszego ruchu, **mogą później być niezgodne**;
- by znaleźć spójną strategię, poszczególne determinizacje trzeba badać **łącznie**;

Problem łączenia strategii [Frank 1996]



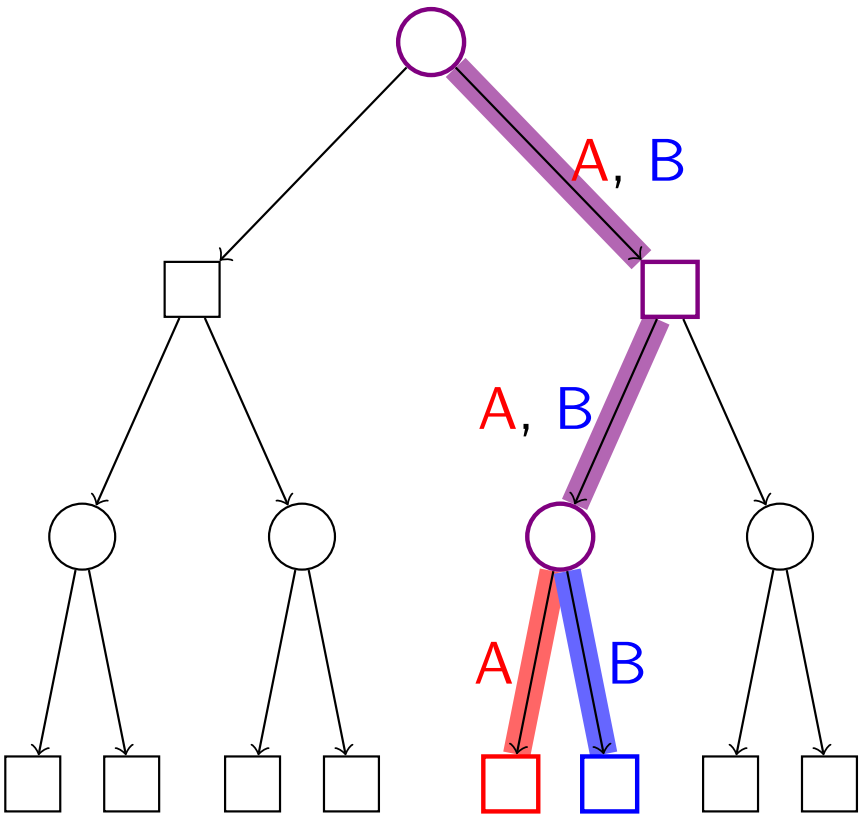
- ruch wygrywający w każdej z pozycji (determinizacji) **z osobna** nie gwarantuje wygranej w grze bez doskonałej informacji;
- znalezione dla poszczególnych determinizacji (np. A i B) zwycięskie strategie, pomimo wspólnego pierwszego ruchu, **mogą później być niezgodne**;
- by znaleźć spójną strategię, poszczególne determinizacje trzeba badać **łącznie**;

Problem łączenia strategii [Frank 1996]



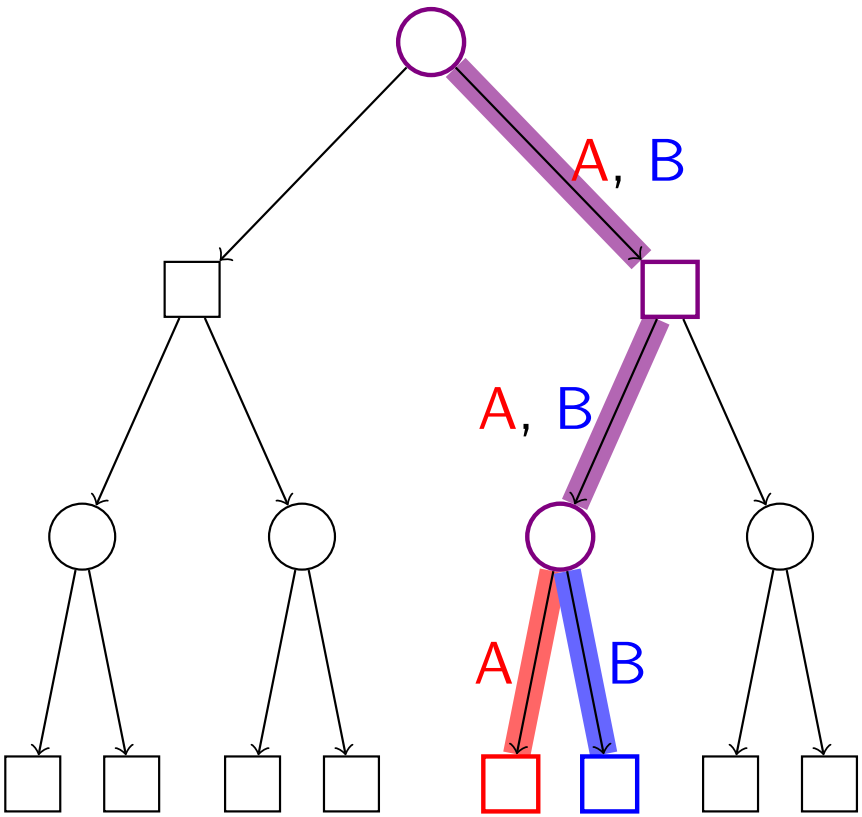
- ruch wygrywający w każdej z pozycji (determinizacji) **z osobna** nie gwarantuje wygranej w grze bez doskonałej informacji;
- znalezione dla poszczególnych determinizacji (np. A i B) zwycięskie strategie, pomimo wspólnego pierwszego ruchu, **mogą później być niezgodne**;
- by znaleźć spójną strategię, poszczególne determinizacje trzeba badać **łącznie**;

Problem łączenia strategii [Frank 1996]



- ruch wygrywający w każdej z pozycji (determinizacji) **z osobna** nie gwarantuje wygranej w grze bez doskonałej informacji;
- znalezione dla poszczególnych determinizacji (np. A i B) zwycięskie strategie, pomimo wspólnego pierwszego ruchu, **mogą później być niezgodne**;
- by znaleźć spójną strategię, poszczególne determinizacje trzeba badać **łącznie**;

Problem łączenia strategii [Frank 1996]



- ruch wygrywający w każdej z pozycji (determinizacji) **z osobna** nie gwarantuje wygranej w grze bez doskonałej informacji;
- znalezione dla poszczególnych determinizacji (np. A i B) zwycięskie strategie, pomimo wspólnego pierwszego ruchu, **mogą później być niezgodne**;
- by znaleźć spójną strategię, poszczególne determinizacje trzeba badać **łącznie**;

MCTS dla gier bez doskonałej informacji

- MCTS (oraz jego odmianę UCT) można zaadaptować do gier bez doskonałej informacji,
- np. można go użyć do niezależnego rozwiązywania determinizacji w algorytmie PIMC,
- jednakże, wykorzystanie jednego, wspólnego drzewa T przy rozpatrywaniu kolejnych determinizacji, pozwala uzyskać silniejszą metodę, odporniejszą na problem łączenia strategii [Schäfer 2008]...

MCTS dla gier bez doskonałej informacji

- MCTS (oraz jego odmianę UCT) można zaadaptować do gier bez doskonałej informacji,
- np. można go użyć do niezależnego rozwiązywania determinizacji w algorytmie PIMC,
- jednakże, wykorzystanie jednego, wspólnego drzewa T przy rozpatrywaniu kolejnych determinizacji, pozwala uzyskać silniejszą metodę, odporniejszą na problem łączenia strategii [Schäfer 2008]...

MCTS dla gier bez doskonałej informacji

- MCTS (oraz jego odmianę UCT) można zaadaptować do gier bez doskonałej informacji,
- np. można go użyć do niezależnego rozwiązywania determinizacji w algorytmie PIMC,
- jednakże, wykorzystanie jednego, wspólnego drzewa T przy rozpatrywaniu kolejnych determinizacji, pozwala uzyskać silniejszą metodę, odporniejszą na problem łączenia strategii [Schäfer 2008]...

MCTS dla gier bez doskonałej informacji – wspólne drzewo

- każdy węzeł w drzewie T reprezentuje zbiór pozycji „nierozróżnialnych" z punktu widzenia gracza Max, przy tym:
- albo ruch we wszystkich pozycjach ze zbioru należy do Maxa i wtedy jest to zbiór informacyjny,
- albo ruch we wszystkich pozycjach ze zbioru należy do Mina i wtedy poszczególne pozycje mogą różnić się możliwymi do wykonania (przez Mina) ruchami,
- to z kolei wymusza modyfikacje MCTSa:
- na etapie selekcji trzeba ograniczyć wybór do zagrań dopuszczalnych w rozpatrywanej determinizacji,
- przeprowadzając ekspansję drzewa T, należy pamiętać o dodaniu następników niedostępnych w aktualnie rozpatrywanej determinizacji;
- w selekcji przez UCT, we wzorze na $f(x)$, liczbę n_w (liczba dotychczasowych rozgrywek, które przebiegły przez w – ojca węzła x), należy zastąpić liczbą dotychczasowych rozgrywek, które przebiegły przez w i w których ruch do x był dostępny.

MCTS dla gier bez doskonałej informacji – wspólne drzewo

- każdy węzeł w drzewie T reprezentuje zbiór pozycji „nierozróżnialnych" z punktu widzenia gracza Max, przy tym:
- albo ruch we wszystkich pozycjach ze zbioru należy do Maxa i wtedy jest to zbiór informacyjny,
- albo ruch we wszystkich pozycjach ze zbioru należy do Mina i wtedy poszczególne pozycje mogą różnić się możliwymi do wykonania (przez Mina) ruchami,
- to z kolei wymusza modyfikacje MCTSa:
- na etapie selekcji trzeba ograniczyć wybór do zagrań dopuszczalnych w rozpatrywanej determinizacji,
- przeprowadzając ekspansję drzewa T, należy pamiętać o dodaniu następników niedostępnych w aktualnie rozpatrywanej determinizacji;
- w selekcji przez UCT, we wzorze na $f(x)$, liczbę n_w (liczba dotychczasowych rozgrywek, które przebiegły przez w – ojca węzła x), należy zastąpić liczbą dotychczasowych rozgrywek, które przebiegły przez w i w których ruch do x był dostępny.

MCTS dla gier bez doskonałej informacji – wspólne drzewo

- każdy węzeł w drzewie T reprezentuje zbiór pozycji „nierozróżnialnych" z punktu widzenia gracza Max, przy tym:
- albo ruch we wszystkich pozycjach ze zbioru należy do Maxa i wtedy jest to zbiór informacyjny,
- albo ruch we wszystkich pozycjach ze zbioru należy do Mina i wtedy poszczególne pozycje mogą różnić się możliwymi do wykonania (przez Mina) ruchami,
- to z kolei wymusza modyfikacje MCTSa:
- na etapie selekcji trzeba ograniczyć wybór do zagrań dopuszczalnych w rozpatrywanej determinizacji,
- przeprowadzając ekspansję drzewa T, należy pamiętać o dodaniu następników niedostępnych w aktualnie rozpatrywanej determinizacji;
- w selekcji przez UCT, we wzorze na $f(x)$, liczbę n_w (liczba dotychczasowych rozgrywek, które przebiegły przez w – ojca węzła x), należy zastąpić liczbą dotychczasowych rozgrywek, które przebiegły przez w i w których ruch do x był dostępny.

MCTS dla gier bez doskonałej informacji – wspólne drzewo

- każdy węzeł w drzewie T reprezentuje zbiór pozycji „nierozróżnialnych" z punktu widzenia gracza Max, przy tym:
- albo ruch we wszystkich pozycjach ze zbioru należy do Maxa i wtedy jest to zbiór informacyjny,
- albo ruch we wszystkich pozycjach ze zbioru należy do Mina i wtedy poszczególne pozycje mogą różnić się możliwymi do wykonania (przez Mina) ruchami,
- to z kolei wymusza modyfikacje MCTSa:
- na etapie selekcji trzeba ograniczyć wybór do zagrań dopuszczalnych w rozpatrywanej determinizacji,
- przeprowadzając ekspansję drzewa T, należy pamiętać o dodaniu następników niedostępnych w aktualnie rozpatrywanej determinizacji;
- w selekcji przez UCT, we wzorze na $f(x)$, liczbę n_w (liczba dotychczasowych rozgrywek, które przebiegły przez w – ojca węzła x), należy zastąpić liczbą dotychczasowych rozgrywek, które przebiegły przez w i w których ruch do x był dostępny.

MCTS dla gier bez doskonałej informacji – wspólne drzewo

- każdy węzeł w drzewie T reprezentuje zbiór pozycji „nierozróżnialnych" z punktu widzenia gracza Max, przy tym:
- albo ruch we wszystkich pozycjach ze zbioru należy do Maxa i wtedy jest to zbiór informacyjny,
- albo ruch we wszystkich pozycjach ze zbioru należy do Mina i wtedy poszczególne pozycje mogą różnić się możliwymi do wykonania (przez Mina) ruchami,
- to z kolei wymusza modyfikacje MCTSa:
- na etapie selekcji trzeba ograniczyć wybór do zagrań dopuszczalnych w rozpatrywanej determinizacji,
- przeprowadzając ekspansję drzewa T, należy pamiętać o dodaniu następników niedostępnych w aktualnie rozpatrywanej determinizacji;
- w selekcji przez UCT, we wzorze na $f(x)$, liczbę n_w (liczba dotychczasowych rozgrywek, które przebiegły przez w – ojca węzła x), należy zastąpić liczbą dotychczasowych rozgrywek, które przebiegły przez w i w których ruch do x był dostępny.

MCTS dla gier bez doskonałej informacji – wspólne drzewo

- każdy węzeł w drzewie T reprezentuje zbiór pozycji „nierozróżnialnych" z punktu widzenia gracza Max, przy tym:
- albo ruch we wszystkich pozycjach ze zbioru należy do Maxa i wtedy jest to zbiór informacyjny,
- albo ruch we wszystkich pozycjach ze zbioru należy do Mina i wtedy poszczególne pozycje mogą różnić się możliwymi do wykonania (przez Mina) ruchami,
- to z kolei wymusza modyfikacje MCTSa:
- na etapie selekcji trzeba ograniczyć wybór do zagrań dopuszczalnych w rozpatrywanej determinizacji,
- przeprowadzając ekspansję drzewa T, należy pamiętać o dodaniu następników niedostępnych w aktualnie rozpatrywanej determinizacji;
- w selekcji przez UCT, we wzorze na $f(x)$, liczbę n_w (liczba dotychczasowych rozgrywek, które przebiegły przez w – ojca węzła x), należy zastąpić liczbą dotychczasowych rozgrywek, które przebiegły przez w i w których ruch do x był dostępny.

MCTS dla gier bez doskonałej informacji – wspólne drzewo

- każdy węzeł w drzewie T reprezentuje zbiór pozycji „nierozróżnialnych" z punktu widzenia gracza Max, przy tym:
- albo ruch we wszystkich pozycjach ze zbioru należy do Maxa i wtedy jest to zbiór informacyjny,
- albo ruch we wszystkich pozycjach ze zbioru należy do Mina i wtedy poszczególne pozycje mogą różnić się możliwymi do wykonania (przez Mina) ruchami,
- to z kolei wymusza modyfikacje MCTSa:
- na etapie selekcji trzeba ograniczyć wybór do zagrań dopuszczalnych w rozpatrywanej determinizacji,
- przeprowadzając ekspansję drzewa T, należy pamiętać o dodaniu następników niedostępnych w aktualnie rozpatrywanej determinizacji;
- w selekcji przez UCT, we wzorze na $f(x)$, liczbę n_w (liczba dotychczasowych rozgrywek, które przebiegły przez w – ojca węzła x), należy zastąpić liczbą dotychczasowych rozgrywek, które przebiegły przez w i w których ruch do x był dostępny.

Problem braku lokalności [Frank 1996]

- w grach z doskonałą informacją wartość pozycji zależy jedynie od podgrafu złożonego z węzłów z niej osiągalnych;
- w grach bez doskonałej informacji, analogiczne stwierdzenie nie jest prawdziwe;
- uczestnicy mogą wyciągać wnioski z zaobserwowanych wcześniej zagrań przeciwnika;
- istotne są nawet te części grafu gry, które nie zostały i nie mogą już zostać osiągnięte;
- rozwiązanie dla PIMC: ważenie poszczególnych determinizacji (względnymi) prawdopodobieństwami tego, że są one bieżącą pozycją w grze.

Problem braku lokalności [Frank 1996]

- w grach z doskonałą informacją wartość pozycji zależy jedynie od podgrafu złożonego z węzłów z niej osiągalnych;
- w grach bez doskonałej informacji, analogiczne stwierdzenie nie jest prawdziwe;
- uczestnicy mogą wyciągać wnioski z zaobserwowanych wcześniej zagrań przeciwnika;
- istotne są nawet te części grafu gry, które nie zostały i nie mogą już zostać osiągnięte;
- rozwiązanie dla PIMC: ważenie poszczególnych determinizacji (względnymi) prawdopodobieństwami tego, że są one bieżącą pozycją w grze.

Problem braku lokalności [Frank 1996]

- w grach z doskonałą informacją wartość pozycji zależy jedynie od podgrafu złożonego z węzłów z niej osiągalnych;
- w grach bez doskonałej informacji, analogiczne stwierdzenie nie jest prawdziwe;
- uczestnicy mogą wyciągać wnioski z zaobserwowanych wcześniej zagrań przeciwnika;
- istotne są nawet te części grafu gry, które nie zostały i nie mogą już zostać osiągnięte;
- rozwiązanie dla PIMC: ważenie poszczególnych determinizacji (względnymi) prawdopodobieństwami tego, że są one bieżącą pozycją w grze.

Problem braku lokalności [Frank 1996]

- w grach z doskonałą informacją wartość pozycji zależy jedynie od podgrafu złożonego z węzłów z niej osiągalnych;
- w grach bez doskonałej informacji, analogiczne stwierdzenie nie jest prawdziwe;
- uczestnicy mogą wyciągać wnioski z zaobserwowanych wcześniej zagrań przeciwnika;
- istotne są nawet te części grafu gry, które nie zostały i nie mogą już zostać osiągnięte;
- rozwiązanie dla PIMC: ważenie poszczególnych determinizacji (względnymi) prawdopodobieństwami tego, że są one bieżącą pozycją w grze.

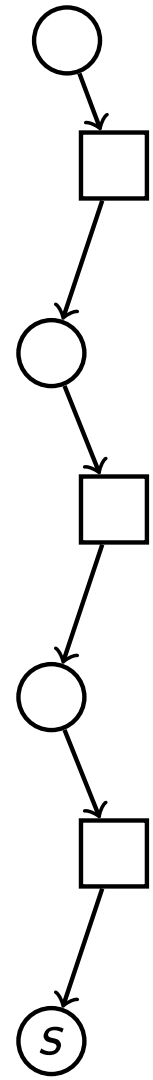
Problem braku lokalności [Frank 1996]

- w grach z doskonałą informacją wartość pozycji zależy jedynie od podgrafu złożonego z węzłów z niej osiągalnych;
- w grach bez doskonałej informacji, analogiczne stwierdzenie nie jest prawdziwe;
- uczestnicy mogą wyciągać wnioski z zaobserwowanych wcześniej zagrań przeciwnika;
- istotne są nawet te części grafu gry, które nie zostały i nie mogą już zostać osiągnięte;
- rozwiązanie dla PIMC: ważenie poszczególnych determinizacji (względnymi) prawdopodobieństwami tego, że są one bieżącą pozycją w grze.

Algorytm szacujący względne prawdopodobieństwo $p(s)$ tego, że gra znajduje się w pozycji s [Beling doktorat]:

1. Zrekonstruuj ścieżkę, przebieg gry od jej początku, do pozycji s.
2. Dla wszystkich pozycji s_i $(i = 1, \ldots, n)$ na tej ścieżce, w których ruch należał do przeciwnika:
   - oszacuj $p_s(s_i)$ – prawdopodobieństwo, że gdyby gra była w pozycji s_i, to przeciwnik wykonałby ruch wzdłuż rozważanej ścieżki,
   - zakładając, że gra on poprawnie;
   - poprawność poszczególnych zagrań przybliż ich poprawnością w determinizacji s_i;
   - użyj wspólnej tablicy transpozycji wykonując obliczenia dla $s_1, \ldots, s_n$ (potem także dla s).
3. Oblicz $p(s) = p_s(s_1)p_s(s_2)\ldots p_s(s_n)$.

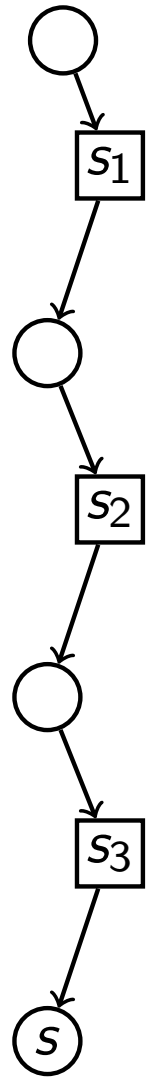
Algorytm szacujący względne prawdopodobieństwo $p(s)$ tego, że gra znajduje się w pozycji s [Beling doktorat]:



1. Zrekonstruuj ścieżkę, przebieg gry od jej początku, do pozycji s.
2. Dla wszystkich pozycji s_i $(i = 1, \ldots, n)$ na tej ścieżce, w których ruch należał do przeciwnika:
   - oszacuj $p_s(s_i)$ – prawdopodobieństwo, że gdyby gra była w pozycji s_i, to przeciwnik wykonałby ruch wzdłuż rozważanej ścieżki,
   - zakładając, że gra on poprawnie;
   - poprawność poszczególnych zagrań przybliż ich poprawnością w determinizacji s_i;
   - użyj wspólnej tablicy transpozycji wykonując obliczenia dla $s_1, \ldots, s_n$ (potem także dla s).
3. Oblicz $p(s) = p_s(s_1)p_s(s_2)\ldots p_s(s_n)$.

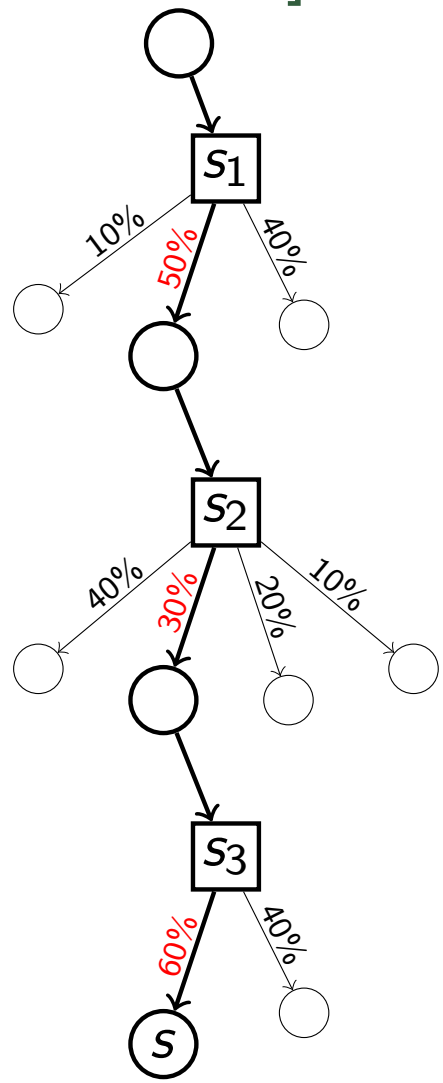
Algorytm szacujący względne prawdopodobieństwo $p(s)$ tego, że gra znajduje się w pozycji s [Beling doktorat]:

1. Zrekonstruuj ścieżkę, przebieg gry od jej początku, do pozycji s.
2. Dla wszystkich pozycji s_i $(i = 1, \ldots, n)$ na tej ścieżce, w których ruch należał do przeciwnika:
   - oszacuj $p_s(s_i)$ – prawdopodobieństwo, że gdyby gra była w pozycji s_i, to przeciwnik wykonałby ruch wzdłuż rozważanej ścieżki,
   - zakładając, że gra on poprawnie;
   - poprawność poszczególnych zagrań przybliż ich poprawnością w determinizacji s_i;
   - użyj wspólnej tablicy transpozycji wykonując obliczenia dla $s_1, \ldots, s_n$ (potem także dla s).
3. Oblicz $p(s) = p_s(s_1)p_s(s_2)\ldots p_s(s_n)$.



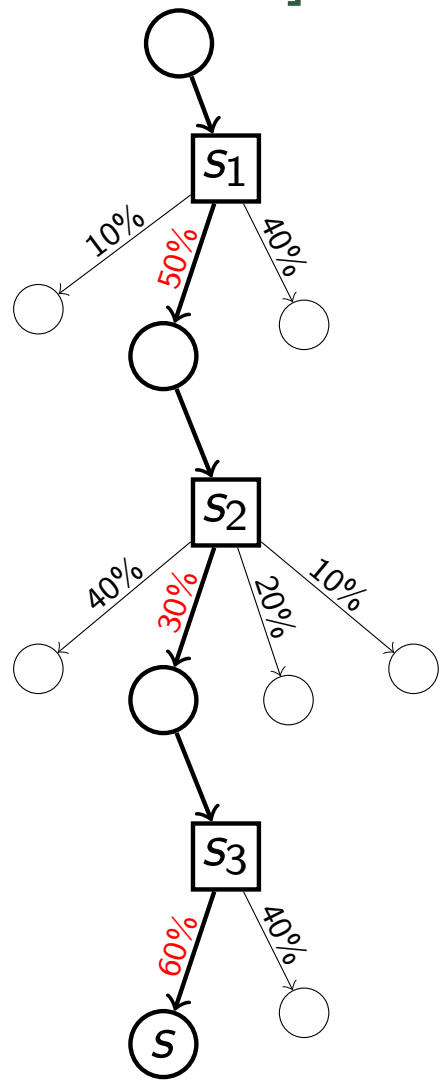
Algorytm szacujący względne prawdopodobieństwo $p(s)$ tego, że gra znajduje się w pozycji s [Beling doktorat]:

1. Zrekonstruuj ścieżkę, przebieg gry od jej początku, do pozycji s.
2. Dla wszystkich pozycji s_i $(i = 1, \ldots, n)$ na tej ścieżce, w których ruch należał do przeciwnika:
   - oszacuj $p_s(s_i)$ – prawdopodobieństwo, że gdyby gra była w pozycji s_i, to przeciwnik wykonałby ruch wzdłuż rozważanej ścieżki,
   - zakładając, że gra on poprawnie;
   - poprawność poszczególnych zagrań przybliż ich poprawnością w determinizacji s_i;
   - użyj wspólnej tablicy transpozycji wykonując obliczenia dla $s_1, \ldots, s_n$ (potem także dla s).
3. Oblicz $p(s) = p_s(s_1)p_s(s_2)\ldots p_s(s_n)$.



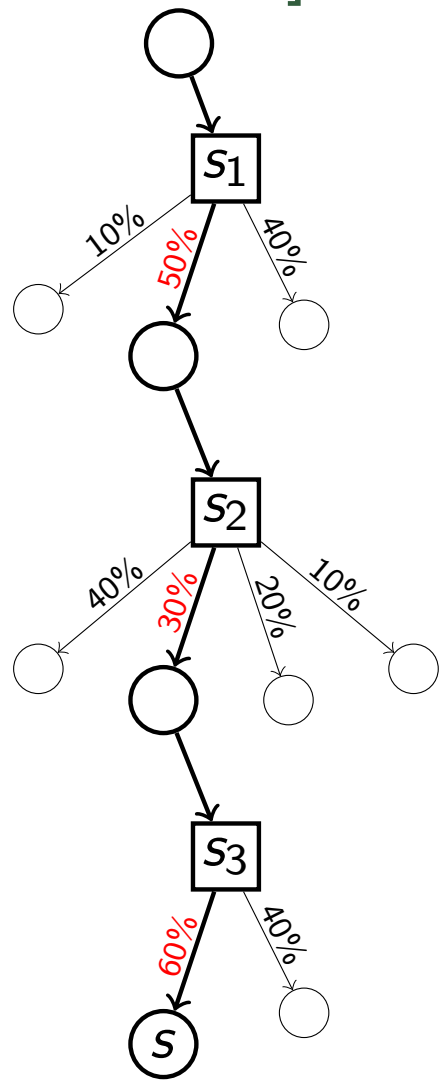
Algorytm szacujący względne prawdopodobieństwo $p(s)$ tego, że gra znajduje się w pozycji s [Beling doktorat]:

1. Zrekonstruuj ścieżkę, przebieg gry od jej początku, do pozycji s.
2. Dla wszystkich pozycji s_i $(i = 1, \ldots, n)$ na tej ścieżce, w których ruch należał do przeciwnika:
   - oszacuj $p_s(s_i)$ – prawdopodobieństwo, że gdyby gra była w pozycji s_i, to przeciwnik wykonałby ruch wzdłuż rozważanej ścieżki,
   - zakładając, że gra on poprawnie;
   - poprawność poszczególnych zagrań przybliż ich poprawnością w determinizacji s_i;
   - użyj wspólnej tablicy transpozycji wykonując obliczenia dla $s_1, \ldots, s_n$ (potem także dla s).
3. Oblicz $p(s) = p_s(s_1)p_s(s_2)\ldots p_s(s_n)$.



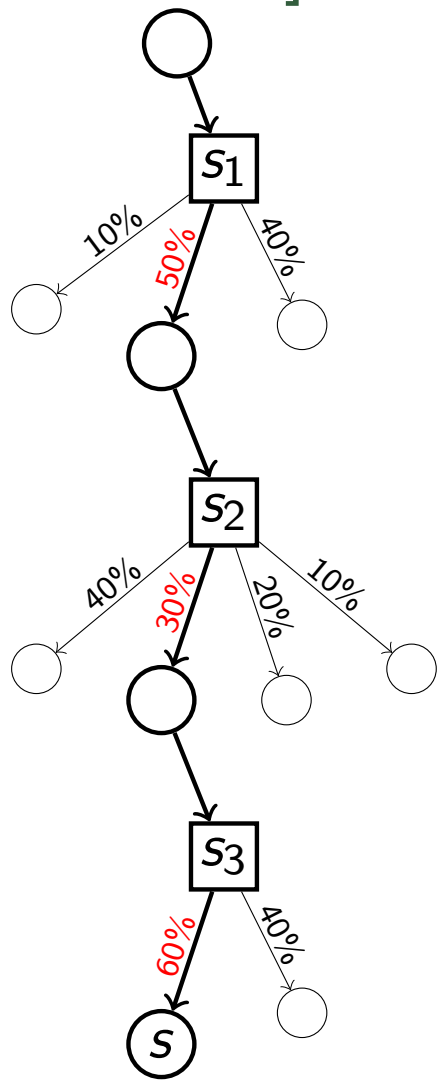
Algorytm szacujący względne prawdopodobieństwo $p(s)$ tego, że gra znajduje się w pozycji s [Beling doktorat]:

1. Zrekonstruuj ścieżkę, przebieg gry od jej początku, do pozycji s.
2. Dla wszystkich pozycji s_i $(i = 1, \ldots, n)$ na tej ścieżce, w których ruch należał do przeciwnika:
   - oszacuj $p_s(s_i)$ – prawdopodobieństwo, że gdyby gra była w pozycji s_i, to przeciwnik wykonałby ruch wzdłuż rozważanej ścieżki,
   - zakładając, że gra on poprawnie;
   - poprawność poszczególnych zagrań przybliż ich poprawnością w determinizacji s_i;
   - użyj wspólnej tablicy transpozycji wykonując obliczenia dla $s_1, \ldots, s_n$ (potem także dla s).
3. Oblicz $p(s) = p_s(s_1)p_s(s_2)\ldots p_s(s_n)$.



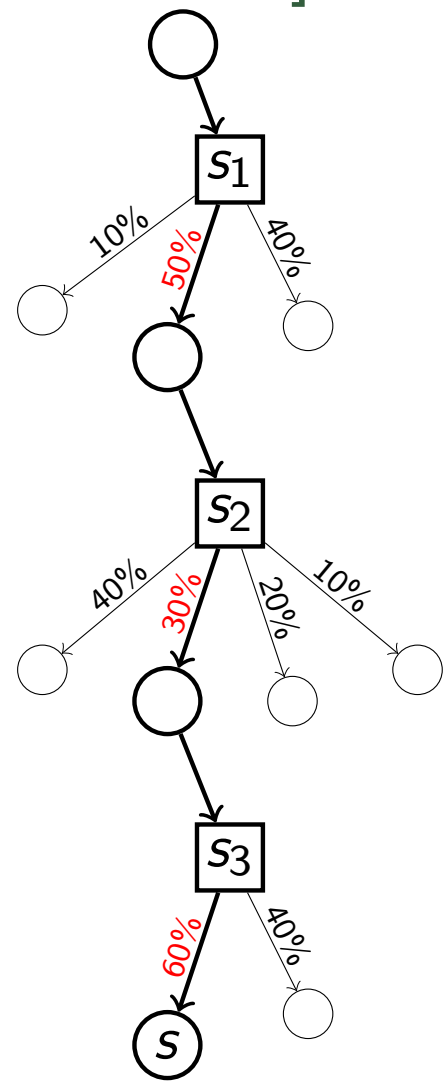
Algorytm szacujący względne prawdopodobieństwo $p(s)$ tego, że gra znajduje się w pozycji s [Beling doktorat]:

1. Zrekonstruuj ścieżkę, przebieg gry od jej początku, do pozycji s.
2. Dla wszystkich pozycji s_i $(i = 1, \ldots, n)$ na tej ścieżce, w których ruch należał do przeciwnika:
   - oszacuj $p_s(s_i)$ – prawdopodobieństwo, że gdyby gra była w pozycji s_i, to przeciwnik wykonałby ruch wzdłuż rozważanej ścieżki,
   - zakładając, że gra on poprawnie;
   - poprawność poszczególnych zagrań przybliż ich poprawnością w determinizacji s_i;
   - użyj wspólnej tablicy transpozycji wykonując obliczenia dla $s_1, \ldots, s_n$ (potem także dla s).
3. Oblicz $p(s) = p_s(s_1)p_s(s_2)\ldots p_s(s_n)$.



Algorytm szacujący względne prawdopodobieństwo $p(s)$ tego, że gra znajduje się w pozycji s [Beling doktorat]:

1. Zrekonstruuj ścieżkę, przebieg gry od jej początku, do pozycji s.
2. Dla wszystkich pozycji s_i $(i = 1, \ldots, n)$ na tej ścieżce, w których ruch należał do przeciwnika:
   - oszacuj $p_s(s_i)$ – prawdopodobieństwo, że gdyby gra była w pozycji s_i, to przeciwnik wykonałby ruch wzdłuż rozważanej ścieżki,
   - zakładając, że gra on poprawnie;
   - poprawność poszczególnych zagrań przybliż ich poprawnością w determinizacji s_i;
   - użyj wspólnej tablicy transpozycji wykonując obliczenia dla $s_1, \ldots, s_n$ (potem także dla s).
3. Oblicz $p(s) = p_s(s_1)p_s(s_2)\ldots p_s(s_n)$.



Bibliografia

- **Louis V. Allis** *Searching for Solutions in Games and Artificial Intelligence*, rozprawa doktorska, 1994
- **H. Jaap van den Herik, et al.** *Games solved: Now and in the future*, Artificial Intelligence 134/2002
- **Donald E. Knuth, Ronald W. Moore** *An Analysis of Alpha-Beta Pruning*, Artificial Intelligence 6/1975
- **L. Kocsis, C. Szepesvári** *Bandit Based Monte-carlo Planning*, Proceedings of the 17th European Conference on Machine Learning, ECML'06, 2006
- **C. Browne, et al.** *A Survey of Monte Carlo Tree Search Methods*, IEEE Transactions on Computational Intelligence and AI in Games, 2012, Volume 4, Issue 1
- **P. Beling** *Szybkie algorytmy decyzyjne w grach z doskonałą informacją i ich wykorzystanie w grach jej pozbawionych*, rozprawa doktorska, Uniwersytet Łódzki, 2016
- **I. Frank** *Search and Planning under Incomplete Information - A Study using Bridge Card Play* rozprawa doktorska, The University of Edinburgh, 1996

Dziękuję za uwagę!