Przeszukiwanie dla gier

Gry są fascynującą rozrywką i często stanowią wyzwanie dla intelektu człowieka. Nic dziwnego, że od dawna były obiektem zainteresowania sztucznej inteligencji.

Metody przeszukiwania w przestrzeni stanów nie dają się bezpośrednio zastosować w typowej grze dwu- lub wieloosobowej ze względu na konieczność uwzględnienia ruchów przeciwnika/ów, które nie są znane. Rozwiązaniem w takim przypadku musi być schemat uwzględniający wszystkie możliwe reakcje przeciwnika/przeciwników.

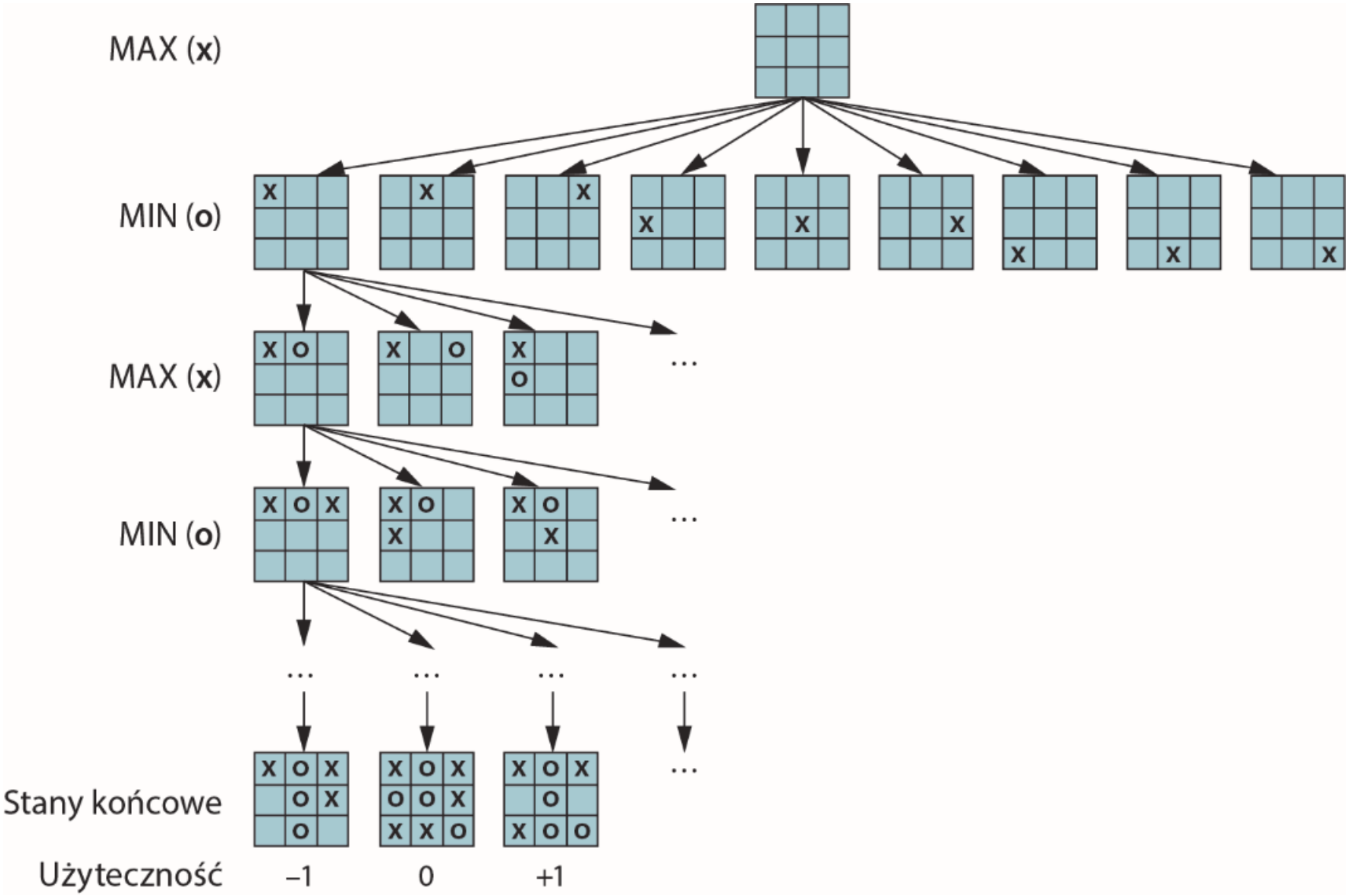
W części gier pełna wiedza o stanie w ogóle nie jest dostępna dla obu graczy. Ponadto, w niektórych grach występuje czynnik losowy (rozdanie kart, rzut kostką, itp.).

Rodzaje gier:

	deterministyczne	losowe
z pełną informacją	szachy, warcaby, go, othello	backgammon, monopol
z niepełną informacją	statki, kółko i krzyżyk na ślepo	brydż, poker, scrabble

Drzewo gry dwuosobowej

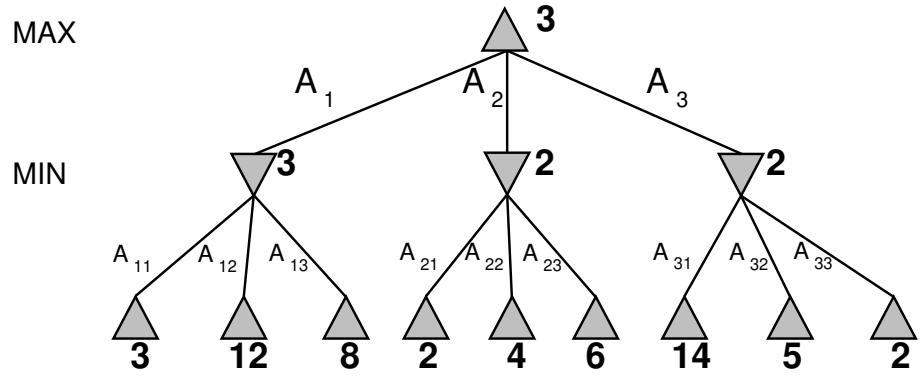
Na początek zajmiemy się grami dwuosobowymi, gdzie zadaniem jest wyznaczenie ruchu jednego z graczy, ale kolejny ruch należy do jego przeciwnika.



Procedura minimaksu

Można wyznaczyć optymalną strategię gry dla gry deterministycznej z pełną informacją za pomocą następującej procedury, zwanej procedurą **minimax**. Oblicza ona wartość węzła startowego przez propagację wartości końcowych (wartości wygranej dla naszego gracza) w górę drzewa gry:

1. poziomy drzewa odpowiadają ruchom graczy: MAX-a i MIN-a; przyjmujemy, że MAX ma pierwszy ruch,
2. stanom terminalnym w liściach drzewa przypisujemy wartość wygranej MAX'a (ujemną, jeśli faktycznie jest to jego przegrana)
3. węzłom drzewa powyżej liści przypisujemy stopniowo wartości: maksymalną ze wszystkich gałęzi jeśli węzeł odpowiada ruchowi MAX-a, i minimalna ze wszystkich gałęzi jeśli węzeł odpowiada ruchowi MIN-a,
4. najwyższa gałąź o największą wartością wskazuje optymalny ruch MAX-a.



Ograniczenie zasobów — zastosowanie heurystyki

Procedura minimaksu definiuje optymalną strategię gracza przy założeniu, że przeciwnik również gra optymalnie. Jednak tylko pod warunkiem, że da się przeanalizować całe drzewo gry.

> Dla prawdziwego drzewa gry może być z tym problem. Np., dla szachów $b \approx 35, m \approx 100$ dla sensownej rozgrywki, i kompletne drzewo gry może mieć około $35^{100} \approx 10^{155}$ węzłów.
> (Liczba atomów w znanej części Wszechświata szacowana jest na 10^{80}.)

Aby rozwiązać ten problem, można posłużyć się **heurystyczną funkcją oceny wartości pozycji**, aby podobnie jak w zwykłych metodach przeszukiwania przestrzeni stanów, wyznaczać dobry ruch bez posiadania jawnej reprezentacji całej przestrzeni. W przypadku gry dwuosobowej pozwoliłoby to zastosować tę samą zasadę minimaksu, ale ograniczyć analizę do kilku ruchów.

> Dla szachów, można taką ocenę obliczyć jako **wartość materialną** posiadanych figur, np. 1 za piona, 3 za wieżę lub gońca, 5 za skoczka, i 9 za hetmana. Dodatkowo można uwzględnić wartość pozycji takich jak „dobre rozstawienie pionów", albo wyższą wartość wieży w końcówce gry, a jeszcze wyższą dwóch wież.

Zastosowanie heurystyk — sytuacje specjalne

Ograniczenie głębokości analizy czasami prowadzi do sytuacji szczególnych, które wymagają nieco zmodyfikowanego podejścia.

Jedną z nich jest zagadnienie **opanowanego zagrożenia**. W niektórych sytuacjach funkcja oceny może sugerować wartości korzystne dla jednego z graczy, ale najbliższe ruchy — poza głębokością uwzględnioną przez funkcję przeszukiwania — nieuchronnie doprowadza do drastycznej zmiany. Rozwiązaniem jest wykrywanie takich niestabilnych sytuacji zagrożenia i pogłębienie przeszukiwania aż do osiągnięcia stanów bardziej stabilnych (*quiescent states*).

Innym problemem jest **problem horyzontu**. Ma on miejsce wtedy, gdy nadchodzi nieuchronne zagrożenie dla jednego z graczy, ale jest on w stanie odsuwać je w czasie wykonując ruchy, które jednak nie rozwiązują problemu.

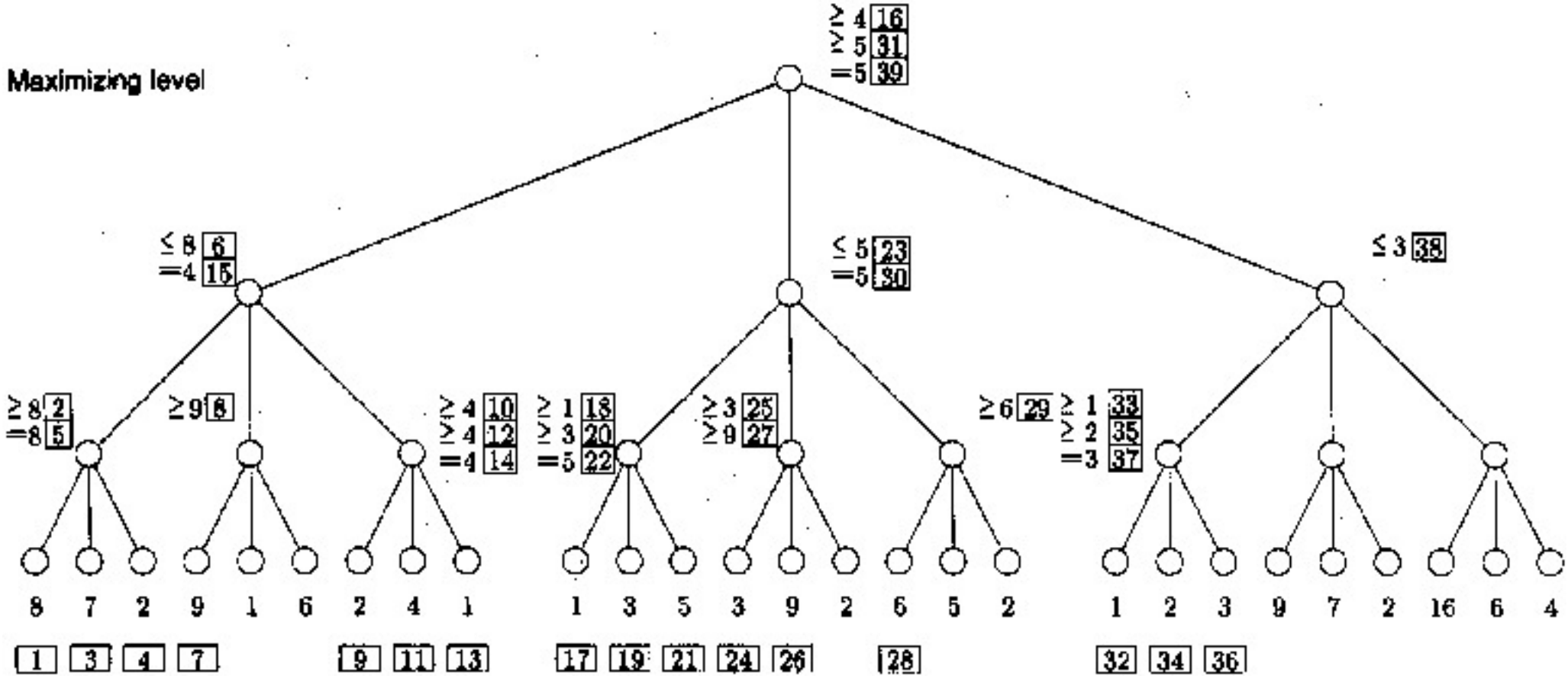
Przeszukiwanie minimax — odcinanie przeszukiwania

Na jakie efekty praktyczne można liczyć stosując ocenę heurystyczną na głębokości kilku ruchów?

Np., dla szachów, przyjmując 10^6 węzłów na sekundę, i 180 sekund na ruch, możemy zbadać $10^8 \approx 35^5$ pozycji, czyli do 5 ruchów wprzód. Program grający w ten sposób zachowuje się racjonalnie, lecz przeciętnemu człowiekowi nietrudno z nim wygrać. Potrzebne są dodatkowe metody zwiększające efektywność przeszukiwania.

Łatwo zauważyć, że można w analizie minimaksowej drzewa gry poczynić pewne oszczędności. Najprostsze z nich nazywane są **odcięciami alfa-beta**.

Odcięcia α–β — przykład



Ćwiczenie: znajdź błąd w powyższym drzewie (źródło: Patrick Henry Winston, *Artificial Intelligence*, 3rd ed.).

Odpowiedź: w kroku 10

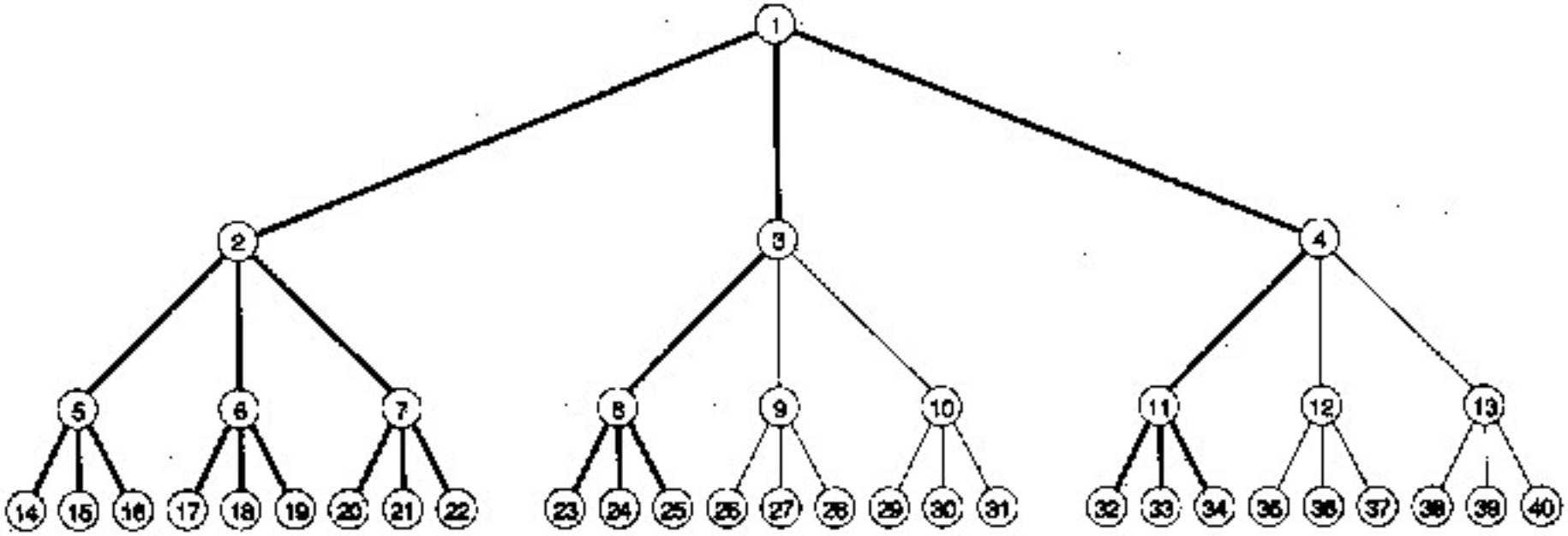
Odcięcia α–β — algorytm

```
PROCEDURE MINIMAX-ALPHA-BETA(n,alpha,beta,depth)
BEGIN
   IF depth==MAXDEPTH THEN RETURN(h(n))

   choices := Lista_potomkow(n)
   WHILE (NOT Empty(choices)) AND (alpha < beta) DO
   ;; zaniechanie badania kolejnych potomkow wezla n oznacza odciecie
   BEGIN
      n1 := First(choices)
      choices := Rest(choices)
      w1 := MINIMAX-ALPHA-BETA(n1,alpha,beta,depth+1)

      IF EVEN(depth) THEN                ; dla wezlow MAX'a
         IF w1 > alpha THEN alpha := w1
      IF ODD(depth) THEN                 ; dla wezlow MIN'a
         IF w1 < beta THEN beta := w1
   END

   IF EVEN(depth) THEN RETURN(alpha)     ; wezel MAX'a
                  ELSE RETURN(beta)      ; wezel MIN'a
END
```

$\Rightarrow$ w pierwszym wywołaniu przyjmujemy $\alpha = -\infty$, $\beta = +\infty$

Ocięcia α–β — przypadek optymalny

Optymalny przypadek przeszukiwania minimaksowego z odcięciami alfa-beta zachodzi gdy na każdym poziomie węzły są rozpatrywane w kolejności od najbardziej korzystnego, dla danego gracza. Wtedy w każdym poddrzewie obliczana jest tylko jedna „seria" węzłów, natomiast przy każdym powrocie w górę drzewa następuje odcięcie.



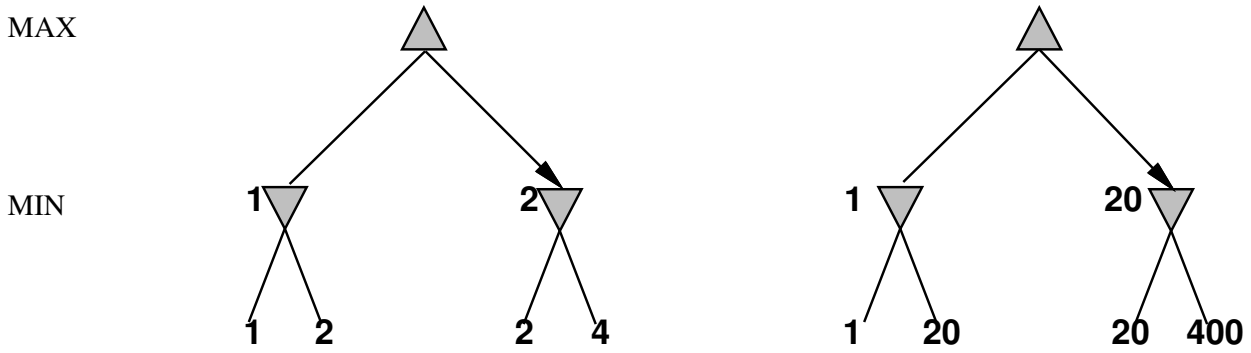
Na powyższym diagramie oszczędność wynosi 16 węzłów; na 27 węzłów na najniższym poziomie obliczonych musi być tylko 11.
Źródło: Patrick Henry Winston, *Artificial Intelligence*, 3rd ed. (uwaga, błąd: węzły 18, 19, 21, i 22 mogłyby również być odcięte).

Własności algorytmu α–β

Zasadniczą ideą algorytmu α–β jest że odcięcia w analizie drzewa nie zmieniają ostatecznego wyniku, tzn. ruchu gracza.

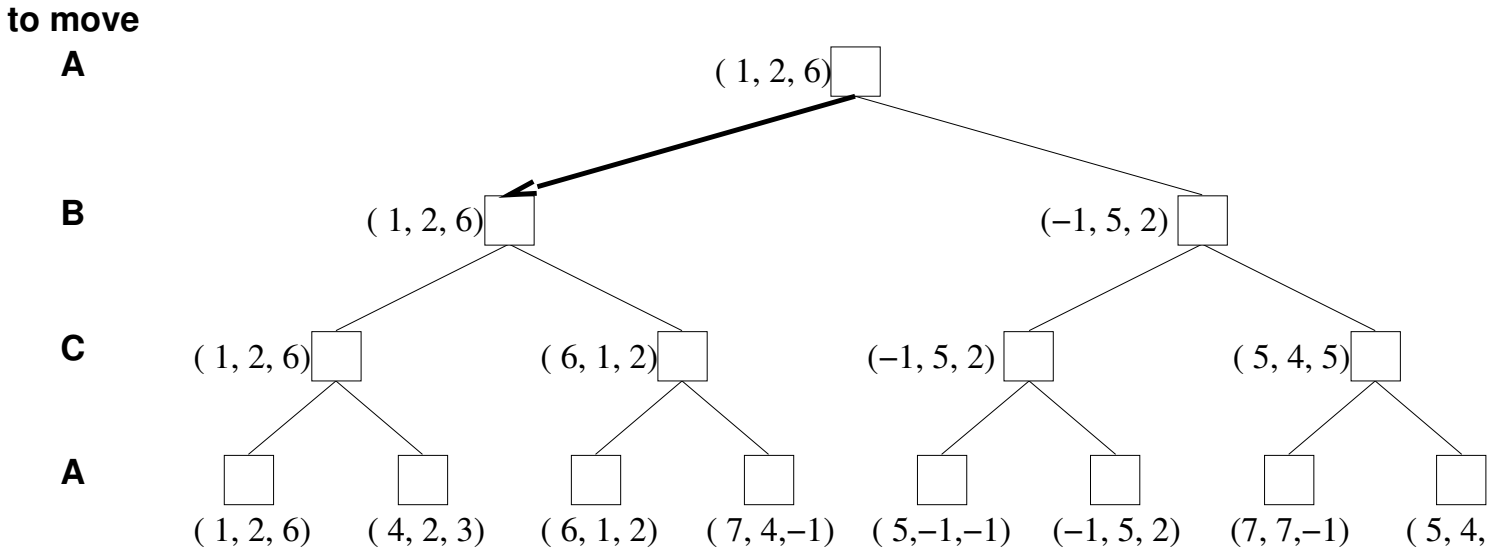
Dobre uporządkowanie pozwala osiągnąć większą efektywność odcinania. W granicy, optymalne odcięcia pozwalają osiągnąć $O(b^{m/2})$.
W praktyce pozwala to podwoić głębokość przeszukiwania.

Zauważmy, że wynik analizy minimax/α–β nie zależy od konkretnych wartości funkcji oceny. Istotne jest tylko uporządkowanie wartości. Oznacza to, że dowolna transformacja monotoniczna funkcji oceny działa tak samo jak oryginalna funkcja.



Minimax — uogólnienie dla gry wieloosobowej

Algorytm minimaksu można łatwo uogólnić na przypadek gry wieloosobowej. Zamiast skalarnej należy zastosować wektorową funkcję oceny, która zwraca wektor ocen wartości pozycji z punktu widzenia poszczególnych graczy. Każdy gracz maksymalizuje swój element wektora, i procedura propagacji oceny w górę drzewa gry przebiega podobnie jak w przypadku gry dwuosobowej.



W grach wieloosobowych pojawiają się jednak dodatkowe elementy strategii takie jak sojusze. Czasami graczom opłaca się zawierać sojusze przeciw innym graczom, a nawet dynamicznie zmieniać te sojusze w trakcie rozgrywki.

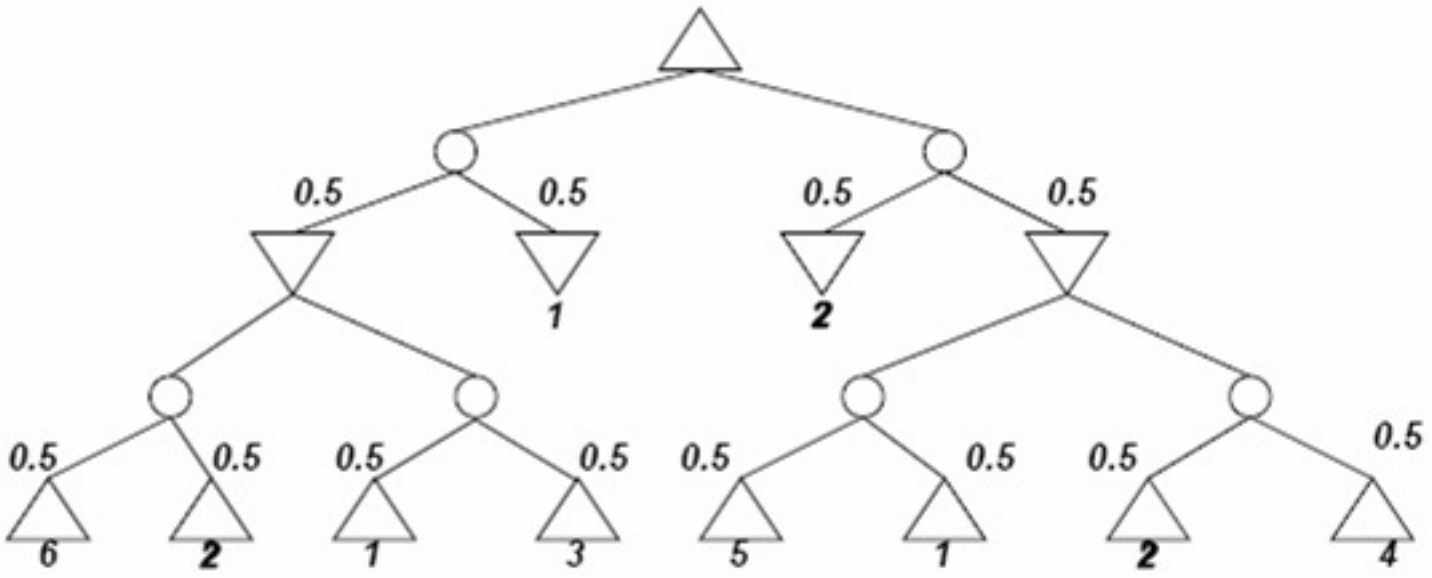
Gry z elementami losowymi — expectimax

Jeżeli w grze występuje czynnik losowy, np. wynik pewnych ruchów zależy od elementu losowego, jak rzut kostką, to analiza takiej gry jest bardziej złożona. Wymaga rozważenia wszystkich możliwości, i obliczania wartości oczekiwanej po dystrybucji zmiennych losowych reprezentujących elementy losowe.

Na przykład, można rozważać gry jednoosobowe z czynnikiem losowym. Co drugi krok jest wyborem gracza, który maksymalizuje wartość swojej funkcji oceny, a co drugi krok jest losowy (lub traktujemy go jako losowy), ze znanym zestawem wyników i znanym rozkładem prawdopodobieństwa tych wyników. Algorytm dostosowany do analizy takich gier nazywa się **expectimax**.

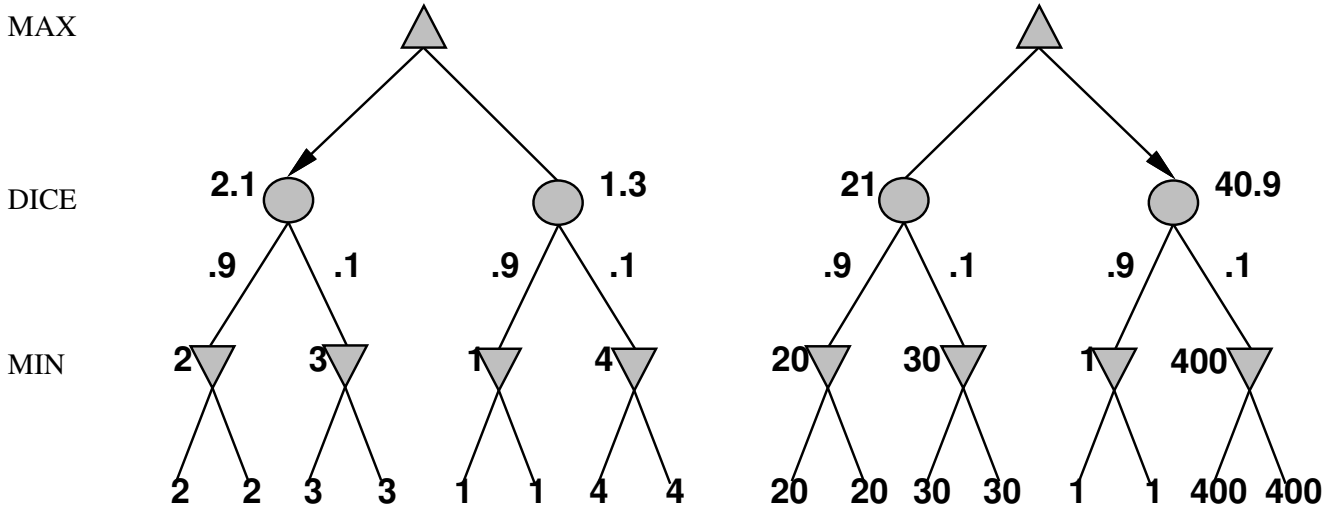
Dalsze uogólnienie — expectiminimax

Pełne uogólnienie algorytmu minimaksu o czynniki losowe uwzględnia ruchy graczy MAXa i MINa oraz kroki losowe. Algorytm analizy drzewa takiej gry nazywa się **expectiminimax**.



Heurystyczna funkcja oceny w expectiminimax

Zauważmy różnicę własności w odniesieniu do stosowanej funkcji oceny. Wybór ruchu na podstawie analizy minimax jest identyczny dla wszystkich funkcji oceny dającej ten sam porządek węzłów. Tej własności nie zachowuje expectiminimax, jak widać na poniższym rysunku. Ruch wybrany dla przedstawionych drzew jest różny, a bez kroku losowego za każdym razem wybrany byłby prawy.



W przypadku expectiminimaksu funkcja oceny nie może być dowolną funkcją poprawnie porządkującą oceniane pozycje. W praktyce musi ona odzwierciedlać oczekiwaną wygraną (lub jej dodatnią liniową transformację).

Gry z niepełną informacją

Przykład: różne gry w karty.

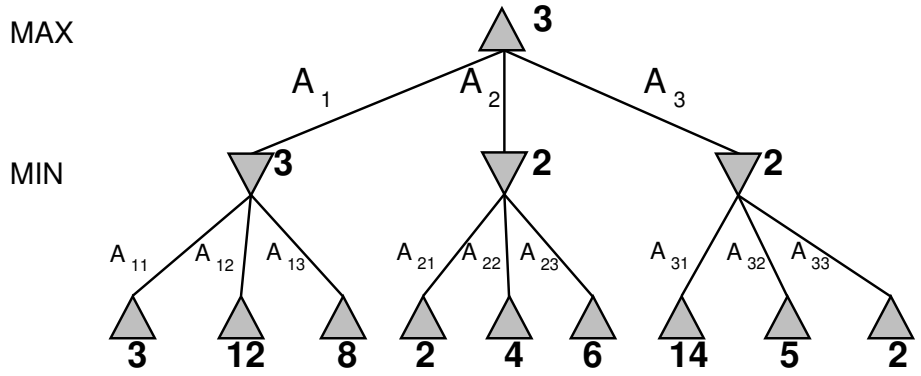
Można obliczyć prawdopodobieństwa wszystkich kombinacji rozdania.

Idea: oblicz wartość minimax każdej możliwej akcji dla każdego rozdania, następnie wybierz akcję z największą wartością oczekiwaną obliczoną dla wszystkich możliwych rozdań.

Najlepsze programy do gry w brydża implementują to podejście przez wygenerowanie wielu rozdań zgodnych z informacją wywnioskowaną z dotychczasowego przebiegu licytacji i rozgrywki, i wybierają akcję, która maksymalizuję liczbę wygranych.

Krótkie podsumowanie — pytania sprawdzające

1. Dla poniższego drzewa gry dwuosobowej, podaj dokładną sekwencję wartości funkcji oceny obliczonych przez algorytm minimaksu z odcięciami alfa/beta (w porządku od lewej do prawej).



Literatura i materiały pomocnicze

1. Stuart Russell, Peter Norvig: Sztuczna inteligencja. Nowe spojrzenie, Wydanie IV, Tom 1, Wyd.Helion, 2023, rozdział 5.