Partition Search i gry z niezupełną informacją

Piotr Butryn

MIMUW
21 stycznia 2010

Definicja

Grą o wartościach w przedziale $[0, 1]$ nazywamy czwórkę $\langle G, p_I, s, ev \rangle$ gdzie:

- G jest skończonym zbiorem legalnych pozycji
- p_I jest pozycją początkową
- $s : G \to 2^G$ jest funkcją zwracająca pozycje możliwe w następnym ruchu
- $ev : G \to \{\mathtt{max}, \mathtt{min}\} \cup [0, 1]$

oraz spełnione są następujące warunki:

- Nie istnieje sekwencja pozycji $p_0, p_1, p_2, \ldots, p_n$ taka, że $p_i \in s(p_{i-1})$ i $p_0 = pn$
- $ev(p) \in [0, 1]$ wtedy i tylko wtedy gdy $s(p) = \emptyset$, tzn. ev przyjmuje wartości liczbowe wtw gdy gra jest zakończona (wartość ev w innych pozycjach mówi nam, który zawodnik wykonuje ruch)

Wartość gry

W dość naturalny sposób możemy zdefiniować funkcję ev_c, mówiącą jaka jest wartość gry w każdej pozycji (nie tylko końcowej):

$$ev_c(p) = \begin{cases} ev(p) & if\ ev(p) \in [0,1] \\ max_{p' \in s(p)} ev_c(p') & if\ ev(p) = \texttt{max} \\ min_{p' \in s(p)} ev_c(p') & if\ ev(p) = \texttt{min} \end{cases}$$

Oznacza to po prostu że gracz MAX wykonuje taki ruch, który skutkuje pozycją o największej wartości ev_c z możliwych, a gracz MIN gra tak, aby osiągnąć pozycję o jak najmniejszej wartości ev_c. Wartością całej gry jest $ev_c(p_I)$

Minimax

Korzystając z definicji funkcji ev_c łatwo stworzyć algorytm wyliczający tę funkcję, a co za tym idzie grający optymalnie w daną grę:

Minimax(p)

if $ev(p) \in [0,1]$ **return** $ev(p)$
if $ev(p) = \texttt{max}$ **return** $max_{p' \in s(p)} minimax(p')$
if $ev(p) = \texttt{min}$ **return** $min_{p' \in s(p)} minimax(p')$

Niestety ten algorytm przeszukuje wszystkie możliwe pozycje, a tych może być BARDZO dużo

Możliwe usprawnienia

Jak łatwo zauważyć przeszukiwanie wszystkich węzłów drzewa nie jest konieczne.
Załóżmy, że sprawdziliśmy już kilka następników dla pozycji p (ruch MAXa) i póki co największa wartość to α. Teraz sprawdzamy $p' \in s(p)$. Trafiamy na $p'' \in s(p')$, dla którego $ev_c(p'') = x \leq \alpha$. Tu możemy zakończyć sprawdzanie, bo gracz MIN na pewno wybierze ten ruch (albo jeszcze lepszy dla siebie), więc $ev_c(p') \leq x \leq \alpha$, a zatem p' nie będzie najkorzystniejszym wyborem dla MAXa (jedynym). Podobnie możemy ograniczyć liczbę sprawdzanych poddrzew w przypadku gracza MIN. Na tych obserwacjach oparty jest algorytm alfa-beta.

Możliwe usprawnienia

Kolejnym usprawnieniem (korzystającym z dodatkowej pamięci) może być stworzenie tablicy pamiętającej wartości wyliczonych już wcześniej pozycji. W wielu grach zdarza się, że zamieniając kolejność ruchów dochodzimy do tej samej pozycji. Dla przykładu, w brydżu zagranie 3 lew (z tymi samymi kartami) możliwe jest czasem na 6^4 sposobów. Po tym zagraniu otrzymujemy zawsze tę samą pozycję i 1000 razy wyliczamy to samo.

Alfa-beta

if there is an entry $(p, [x, y], z)$ in T **return** z
if $\mathtt{ev}(p) \in [0, 1]$ **then** $v_{\text{ans}} = \mathtt{ev}(p)$
if $\mathtt{ev}(p) = \mathtt{max}$ **then**
  $v_{\text{ans}} := 0$
  for each $p' \in s(p)$ **do**
    $v_{\text{new}} = \alpha\beta(p', [\max(v_{\text{ans}}, x), y])$
    if $v_{\text{new}} \geq y$ **then**
      $T := T \cup (p, [x, y], v_{\text{new}})$
      return v_{new}
    if $v_{\text{new}} > v_{\text{ans}}$ **then** $v_{\text{ans}} = v_{\text{new}}$

Alfa-beta

if $\mathtt{ev}(p) = \mathtt{min}$ **then**
  $v_{\text{ans}} := 1$
  for each $p' \in s(p)$ **do**
    $v_{\text{new}} = \alpha\beta(p', [x, \min(v_{\text{ans}}, y)])$
    if $v_{\text{new}} \leq x$ **then**
      $T := T \cup (p, [x, y], v_{\text{new}})$
      return v_{new}
    if $v_{\text{new}} < v_{\text{ans}}$ **then** $v_{\text{ans}} = v_{\text{new}}$
$T := T \cup (p, [x, y], v_{\text{ans}})$
return v_{ans}

Skuteczność

Algorytm alfa-beta redukuje dość mocno średni stopień rozgałęzienia drzewa:

- $b \longrightarrow b^{0.75}$ dla losowego ustawienia potomków
- $b \longrightarrow b^{0.50}$ dla optymalnego ustawienia potomków

Grać jak człowiek

Komputery dysponują mocą obliczeniową nieporównywalnie większą niż człowiek. Ich problemem w wielu grach jest natomiast brak inteligencji. Marnują one mnóstwo czasu na przeliczanie wariantów, które przeciętny gracz od razu by odrzucił.
Przykładem mogą być np. szachy, w których, gdy ktoś zaatakuje naszego hetmana, to próbujemy nim uciekać lub zasłonić się figurą.

- do rozważenia mamy kilka, maksymalnie kilkanaście ruchów
- komputer rozważa natomiast wszystkie możliwe zagrania (do któregoś poziomu)
- nie jest w stanie przełożyć przewagi w mocy na przewagę w sile gry

Algorytmy takie jak alfa-beta niewiele tu pomagają.

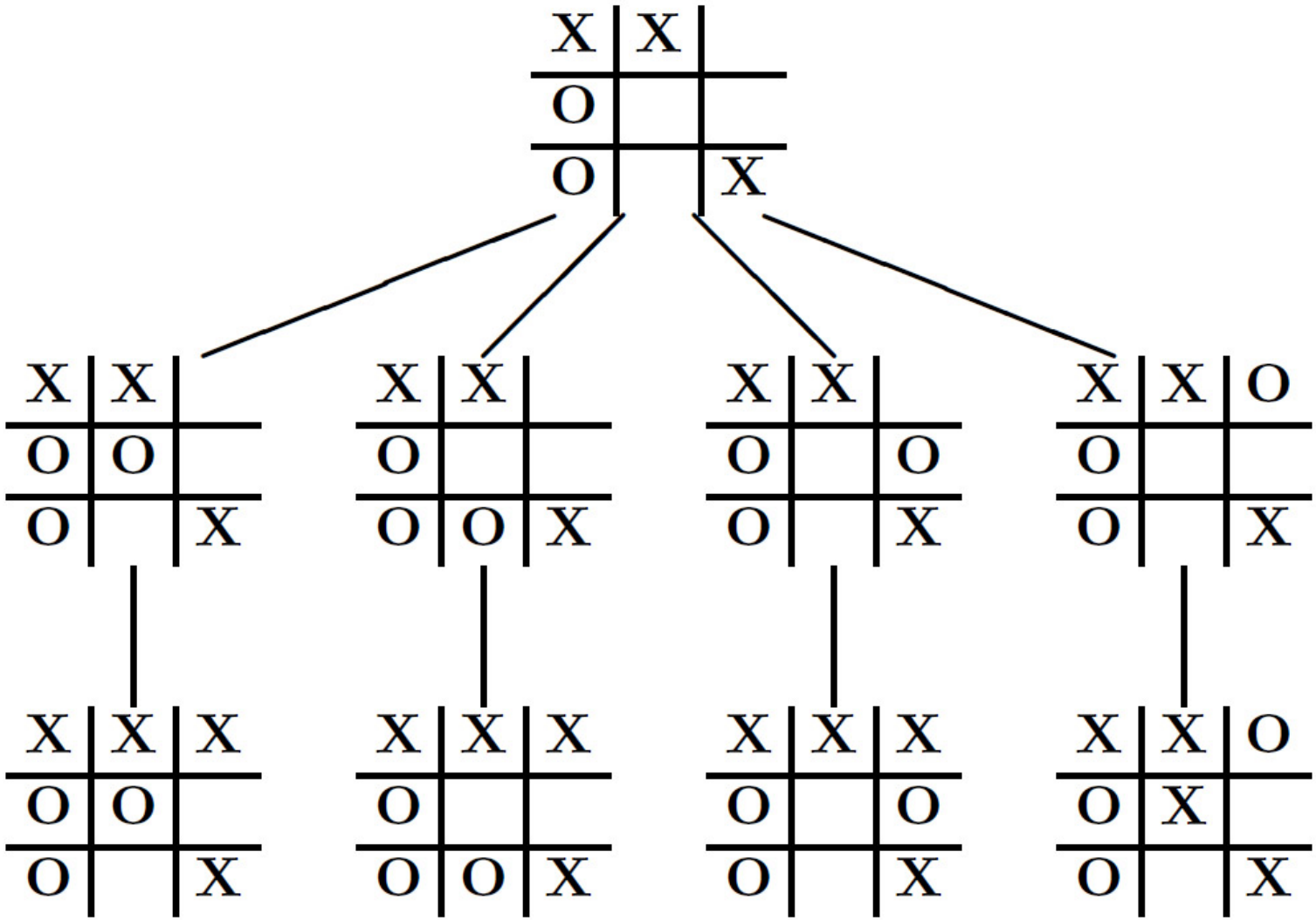
Grać jak człowiek

Drugą cechą wielu gier, na które proste algorytmy nie zwracają uwagi jest lokalność:

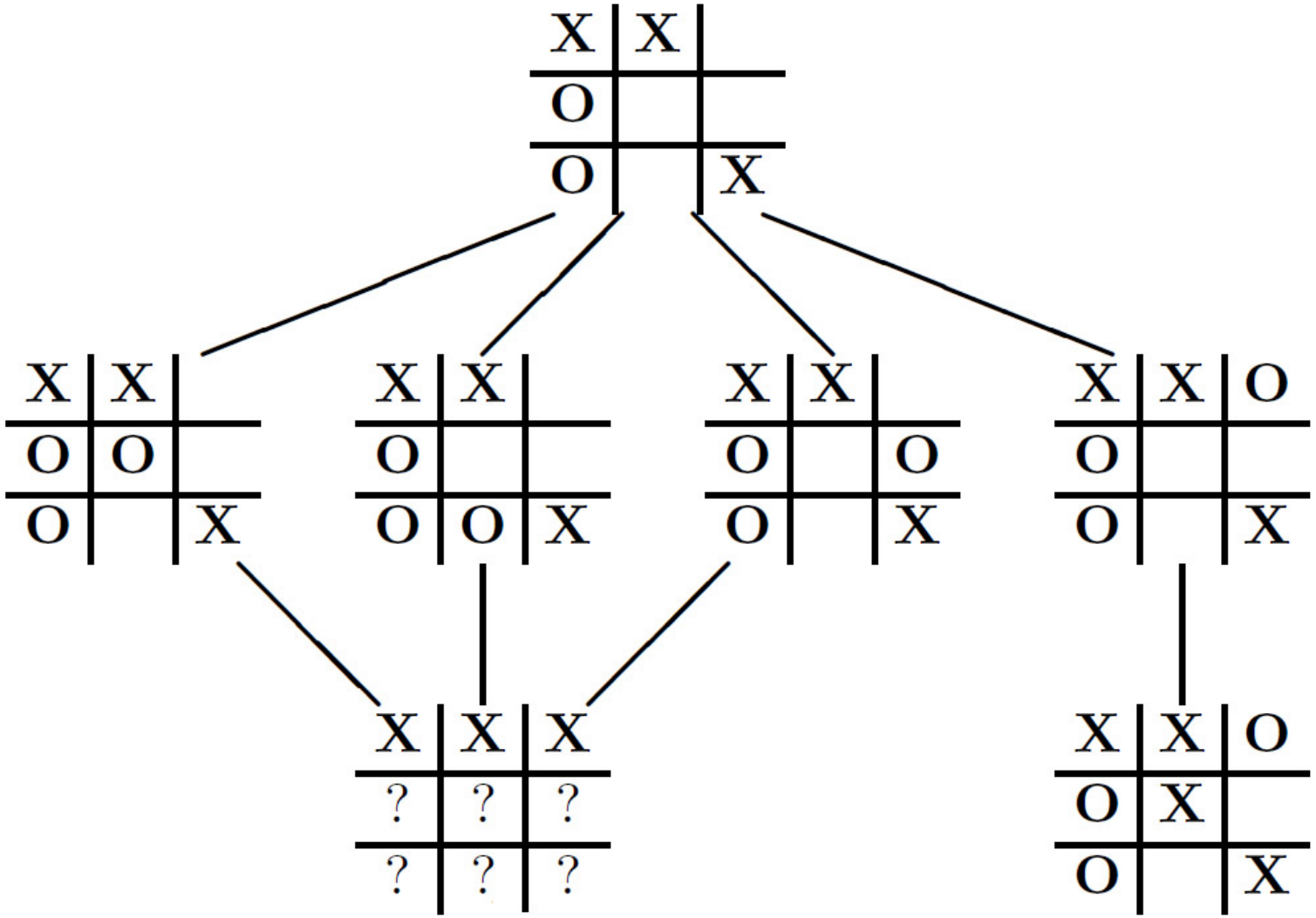
- mat jest najczęściej zasługą tylko kilku figur
- reszta mogłaby stać w innych miejscach lub mogłoby ich wcale nie być
- prawie zawsze moglibyśmy także dostawić na planszę kilka figur, a mat byłby dalej

Dość naturalnie zatem jest zamiast pojedynczych pozycji pamiętać całe ich grupy, które mają tę samą wartość i sprawdzać czy rozpatrywana pozycja pasuje do któregoś ze wzorców.

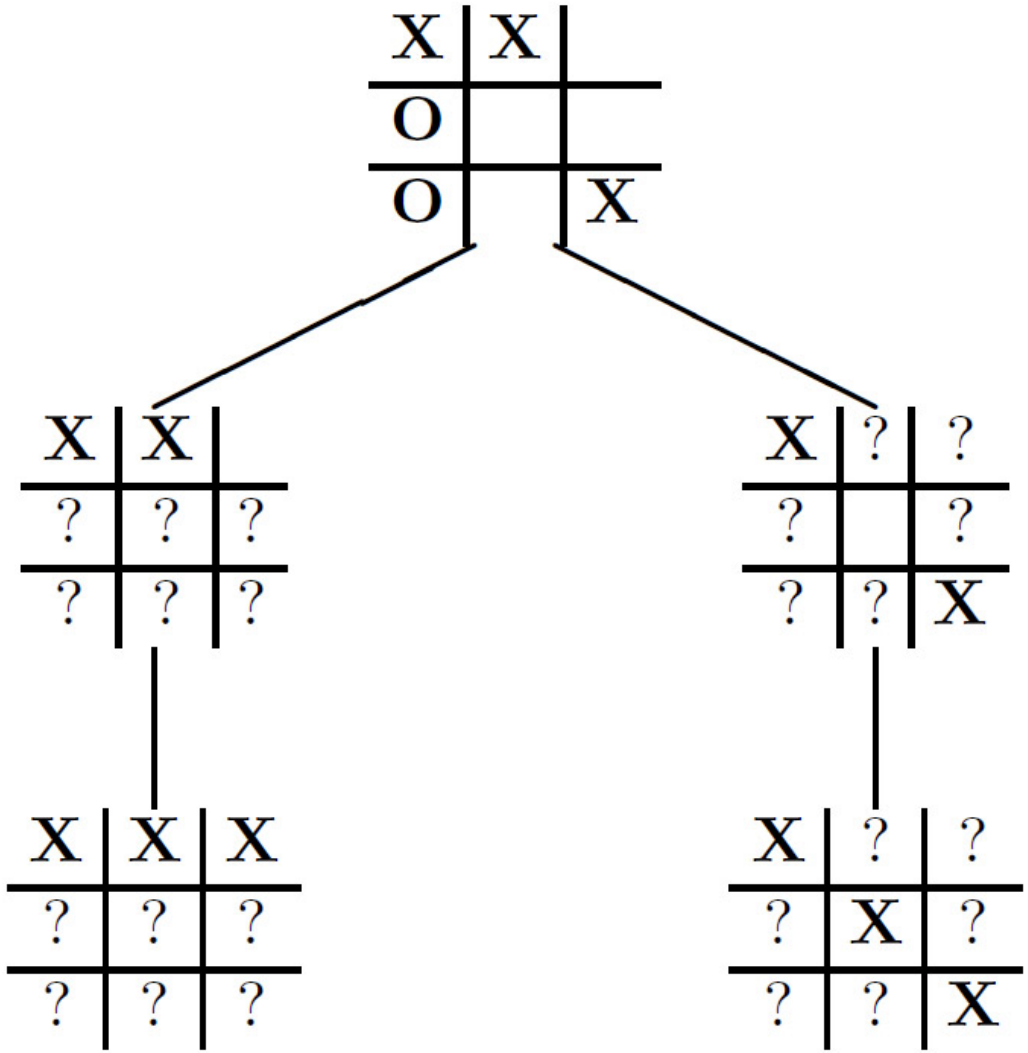
Kółko i krzyżyk



Kółko i krzyżyk



Kółko i krzyżyk



Rozpatrzmy grę $\langle G, p_I, s, ev \rangle$ oraz zbiór pozycji $S \subseteq G$. Powiemy, że zbiorem pozycji **osiągających** S jest zbiór takich pozycji p, że $s(p) \cap S \neq \emptyset$ i będziemy go oznaczać jako $R_0(S)$. Zbiór pozycji p takich, że $s(p) \subseteq S$ będziemy natomiast nazywać zbiorem pozycji **przechodzących** do S i oznaczać $C_0(S)$.
W algorytmie będziemy korzystać z pewnych przybliżeń obu tych zbiorów.

Załóżmy teraz, że istnieje pewien zbiór pozycji S wygrywający dla gracza MAX. To oznacza, że wygrywające są dla niego wszystkie pozycje z $C_0(S)$ oraz te pozycje $p \in R_0(S)$ gdzie $ev(p) = \texttt{max}$.

W naszym algorytmie będziemy konstruować funkcje $R(p, S)$ i $C(p, S)$, które będą reprezentowały zbiory pozycji podobnych do p w sensie osiągania lub przechodzenia do zbioru pozycji wygrywających S.

Rozpatrzmy grę $\langle G, p_I, s, ev \rangle$ oraz funkcję f o zbiorze wartości 2^G (podzbiory G). Powiemy, że f jest **zgodna** z funkcją ev jeśli dla każdych $p, p' \in F$, F w zbiorze wartości f, mamy $ev(p) = ev(p')$. Systemem częściowym dla gry jest trójka $\langle P, R, C \rangle$ funkcji zgodnych z ev takich, że:

- $P : G \to 2^G$ przekształca pozycje w zbiory pozycji ($p \in P(p)$)
- $R : G \times 2^G \to 2^G$ taka, że jeśli $p \in R_0(S)$ to $p \in R(p, S) \subseteq R_0(S)$
- $C : G \times 2^G \to 2^G$ taka, że jeśli $p \in C_0(S)$ to $p \in C(p, S) \subseteq C_0(S)$

Partition Search

if there is an entry $(S, [x, y], z)$ with $p \in S$ **return** $\langle z, S \rangle$
if $\mathtt{ev}(p) \in [0, 1]$ **then** $\langle v_{\text{ans}}, S_{\text{ans}} \rangle = \langle \mathtt{ev}(p), P(p) \rangle$
if $\mathtt{ev}(p) = \mathtt{max}$ **then**
  $v_{\text{ans}} := 0$
  $S_{\text{all}} := \emptyset$
  for each $p' \in s(p)$ **do**
    $\langle v_{\text{new}}, S_{\text{new}} \rangle = \alpha\beta(p', [\max(v_{\text{ans}}, x), y])$
    if $v_{\text{new}} \geq y$ **then**
      $T := T \cup (S_{\text{new}}, [x, y], v_{\text{new}})$
      return $\langle v_{\text{new}}, S_{\text{new}} \rangle$
    if $v_{\text{new}} > v_{\text{ans}}$ **then** $\langle v_{\text{ans}}, S_{\text{ans}} \rangle = \langle v_{\text{new}}, S_{\text{new}} \rangle$
    $S_{\text{all}} := S_{\text{all}} \cup S_{\text{new}}$
  if $v_{\text{ans}} = 0$ **then** $S_{\text{ans}} = C(p, S_{\text{all}})$ †
  else $S_{\text{ans}} = R(p, S_{\text{ans}}) \cap C(p, S_{\text{all}})$ † ‡

Partition Search

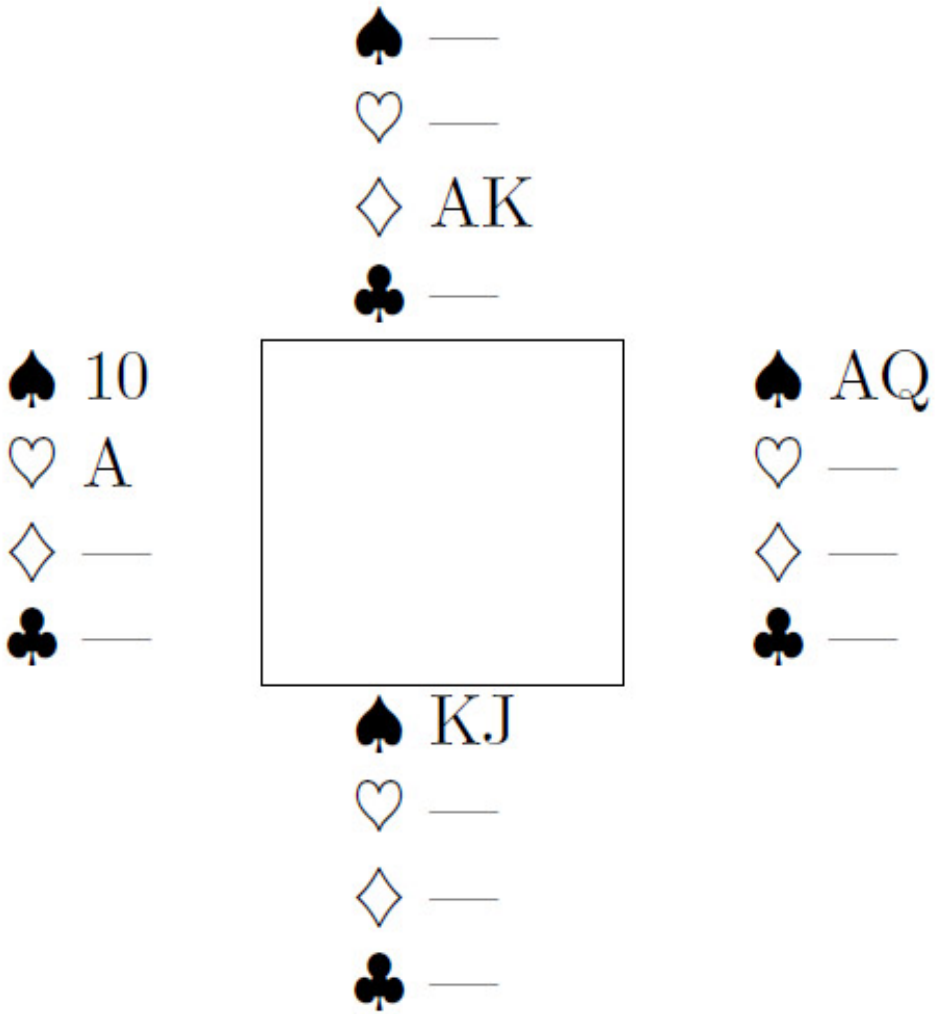
if $\mathtt{ev}(p) = \mathtt{min}$ **then**
  $v_{\text{ans}} := 1$
  $S_{\text{all}} := \emptyset$
  for each $p' \in s(p)$ **do**
    $\langle v_{\text{new}}, S_{\text{new}} \rangle = \alpha\beta(p', [x, \min(v_{\text{ans}}, y)])$
    if $v_{\text{new}} \leq x$ **then**
      $T := T \cup (S_{\text{new}}, [x, y], v_{\text{new}})$
      return $\langle v_{\text{new}}, S_{\text{new}} \rangle$
    if $v_{\text{new}} < v_{\text{ans}}$ **then** $\langle v_{\text{ans}}, S_{\text{ans}} \rangle = \langle v_{\text{new}}, S_{\text{new}} \rangle$
    $S_{\text{all}} := S_{\text{all}} \cup S_{\text{new}}$
  if $v_{\text{ans}} = 1$ **then** $S_{\text{ans}} = C(p, S_{\text{all}})$
  else $S_{\text{ans}} = R(p, S_{\text{ans}}) \cap C(p, S_{\text{all}})$ ‡
$T := T \cup (S_{\text{ans}}, [x, y], v_{\text{ans}})$
return $\langle v_{\text{ans}}, S_{\text{ans}} \rangle$

Poprawność

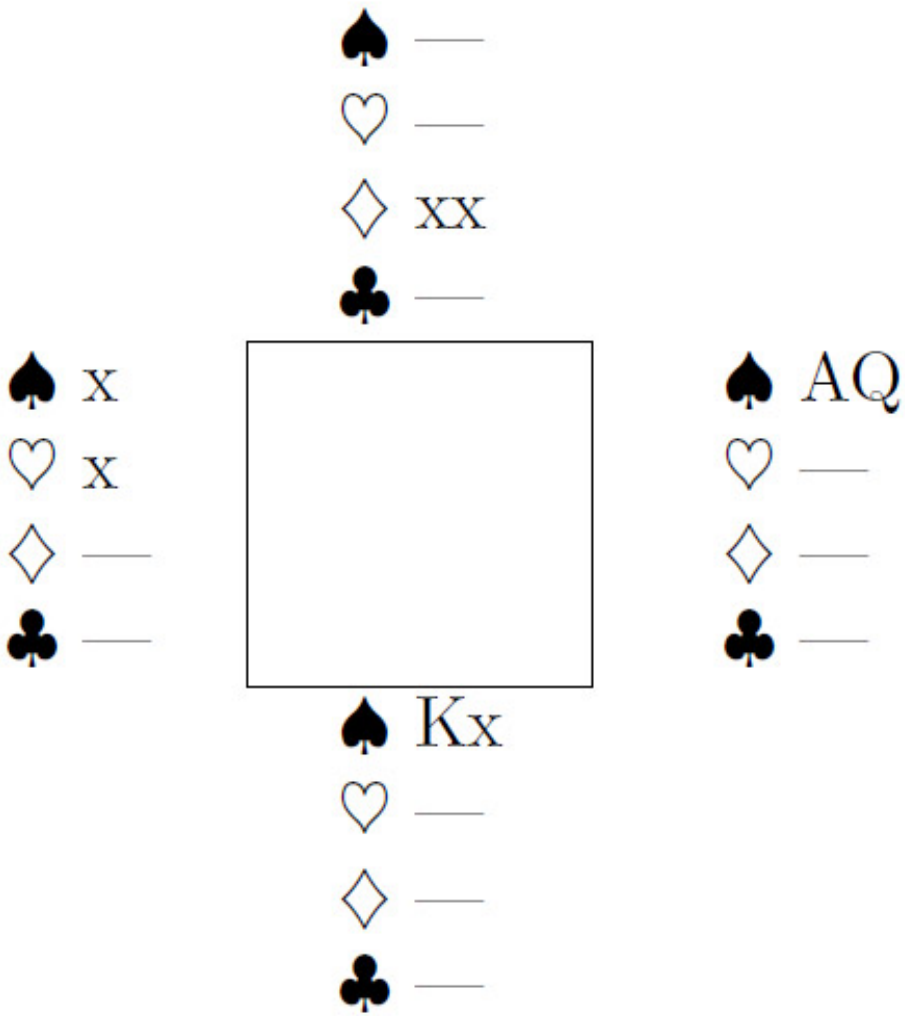
Musimy pokazać, że każda pozycja w S_{ans} ma wartość v_{ans}. Rozpatrzmy przypadek kiedy w węźle gra MAX.

- wartość węzła to 0 - gracz MAX nie może wyjść poza zbiór pozycji przegrywających, a zatem każdy zbiór S_{new} jest przegrywający, czyli także $S_{all} = \cup S_{new}$ jest przegrywający. Zatem za S_{ans}możemy przyjąć $C(p, S_{all})$.
- jeśli wartość większa od zera, to aby inna pozycja p' miała na pewno tę samą wartość, to p' musi osiągać najlepszą możliwość w p, a także musi przechodzić tylko do tych pozycji co p. A zatem musi należeć do $R_0(S_{ans}) \cap C_0(S_{all})$.

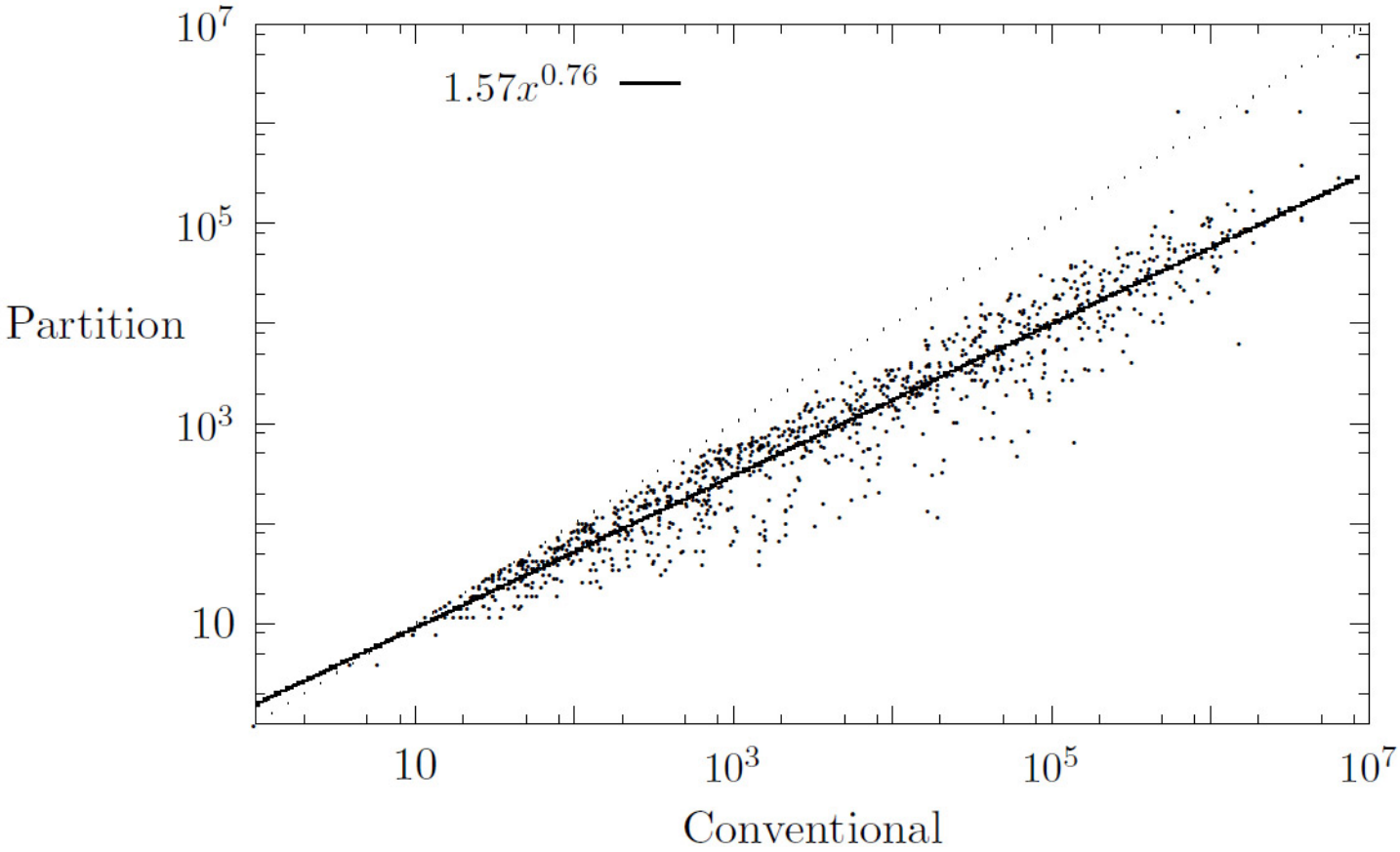
Brydż



Brydż



Brydż


10^7
10^5
10^3
10
Partition
$1.57x^{0.76}$
10
10^3
10^5
10^7
Conventional

Monte Carlo

Szybkie analizowanie w otwarte karty znalazło zastosowanie w programach grających w brydża. Polegają one na wylosowaniu wielu (ok. 100) różnych możliwych kart przeciwników i analizowaniu rozdań w otwarte karty:

1. wylosuj zbiór D rozdań zgadzających się z licytacją i grą do tej pory
2. dla każdego możliwego ruchu m i rozdania d policz wynik rozdania w otwarte karty $(w(m, d))$
3. zwróć ruch, dla którego $\sum_m w(m, d)$ jest maksymalna

Problemy

Niestety takie podejście ma sporo wad:

- brak rozgrywek wywiadowczych
- wyrzucanie potrzebnych kart
- często wybór nieoptymalnej rozgrywki:
  1. rozgrywka wygrywa przy $D\spadesuit$ u W
  2. rozgrywka wygrywa przy $D\spadesuit$ u E
  3. rozgrywka wygrywa przy $D\spadesuit$ u dowolnego gracza
  4. rozgrywka odkłada decyzję na później (ale trzeba będzie wybrać pomiędzy 1 i 2)

Inne podejście

Przyjrzyjmy się wypowiedziom graczy:

- wygrałbym jakby $D\spadesuit$ była u W
- wygrałbym jakby E miał $K\diamondsuit$
- wygrałbym jakby przy 3 kierach były 2 trefle

Jak widać dla gracza z pozycją nie jest powiązana konkretna wartość, lecz funkcja z możliwych układów rąk przeciwników w wynik (dla uproszczenia przyjmijmy, że albo wygra albo przegra) Zatem z pozycją jest związana $f : S \rightarrow \{0, 1\}$. Możemy także zdefiniować $max(f, g)(s) = max(f(s), g(s))$ oraz analogicznie minimum.

Formalnie

Grą nazywamy $\langle G, V, p_I, s, ev, f_+, f_- \rangle$ gdzie:

- G jest skończonym zbiorem pozycji
- V jest zbiorem wartości gry
- $p_I \in G$ jest pozycją początkową
- $s : G \to 2^G$ zwraca zbiór następników
- $ev : G \to \{\mathtt{max}, \mathtt{min}\} \cup V$ zwraca wartość pozycji dla pozycji końcowych oraz określa kto gra w pozostałych
- $f_+ : \mathcal{P}(V) \to V$ i $f_- : \mathcal{P}(V) \to V$ są funkcjami dla gracza MAX i MIN

oraz spełnione są warunki takie jak wcześniej.

Formalnie

Podobnie jak poprzednio możemy zdefiniować funkcję ev_c mówiącą jaka jest wartość gry w pozycjach innych niż końcowe:

$$ev_c(p) = \begin{cases} ev(p) & \text{if } ev(p) \in V \\ f_+\{ev_c(p') \mid p' \in s(p)\} & \text{if } ev(p) = \texttt{max} \\ f_-\{ev_c(p') \mid p' \in s(p)\} & \text{if } ev(p) = \texttt{min} \end{cases}$$

Wartością całej gry jest oczywiście $ev_c(p_I)$.

Formalnie

W naszym przypadku:

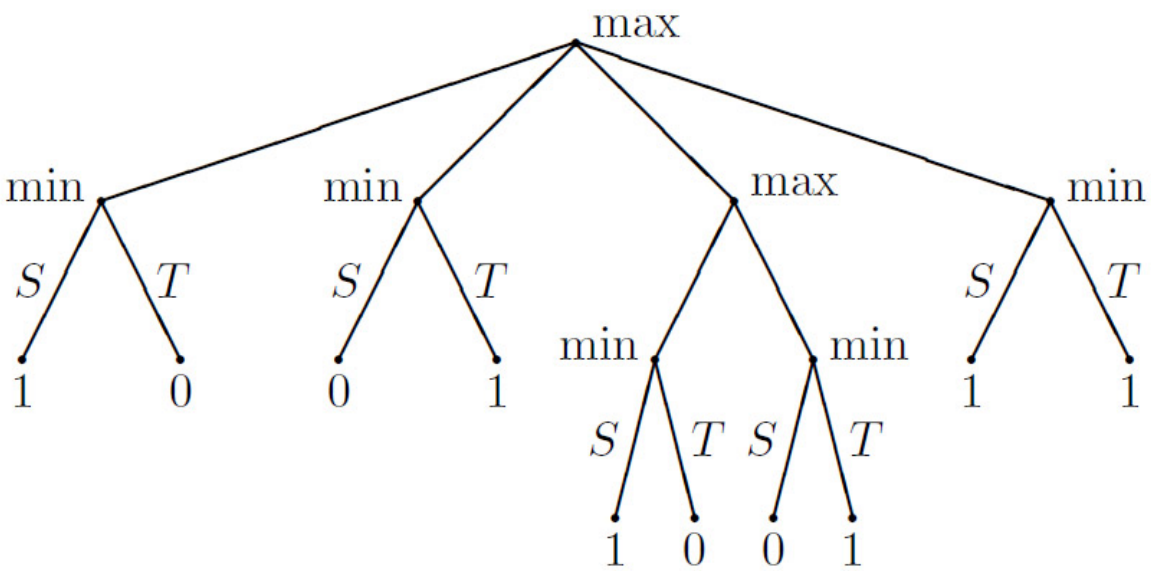
- zbiór G jest zbiorem par (p, Z) gdzie p jest układem kart znanych, a $Z \subset S$ jest zbiorem możliwych układów kart zakrytych zgodnych z dotychczasowymi zagraniami
- zbiór V jest równy 2^S
- $p_I = (p_0, S)$ gdzie p_0 jest układem znanych kart
- funkcja s jest zdefiniowana następująco:
  - jeśli gra MAX, to zagrywa jedną z możliwych kart a Z się nie zmienia
  - jeśli gra MIN, to zagrywa jedną z możliwych kart a Z' jest podzbiorem Z, dla którego zagranie tej karty jest możliwe
- pozycje końcowe, to te kiedy zostaną zagrane wszystkie karty. Wtedy $Z = \{s\}$, bo znamy już położenie wszystkich kart. Wartością gry jest S jeśli wygraliśmy oraz $S - \{s\}$ gdy przegraliśmy
- $f_+(U, V) = U \cup V$ oraz $f_-(U, V) = U \cap V$

Poprawność

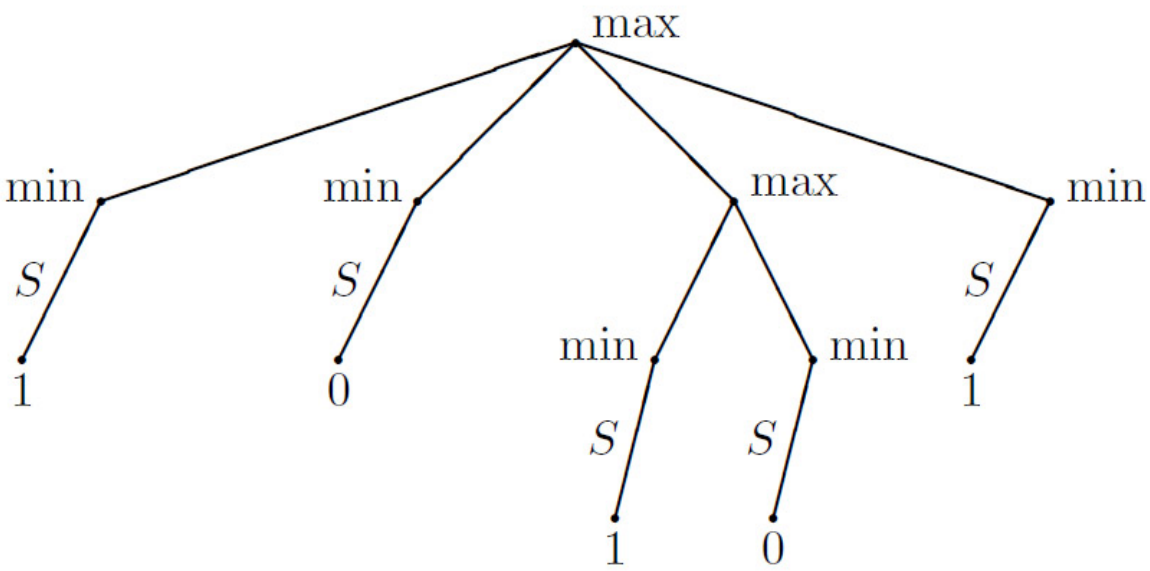
Powinniśmy pokazać, że jeśli w pozycji (p, S) możemy wygrać dla $T \subseteq S$, to wartością gry w (p, S) jest T

- jak korzeń jest końcowy to OK
- (dla MAX) jeśli dla konkretnego układu kart $s \in S$ wygrywamy, to znaczy, że wygrywamy w jakimś następniku (p_i, S_i). Zatem $s \in U_i = ev_c(p_i, S_i)$, czyli należy też do $\bigcup U_i$
- jeśli dla s przegrywamy to nie może on należeć do żadnego U_i, a zatem także do sumy
- dla MIN analogicznie (jak do wszystkich U_i to do przecięcia, jak nie do wszystkich to także nie do przecięcia)

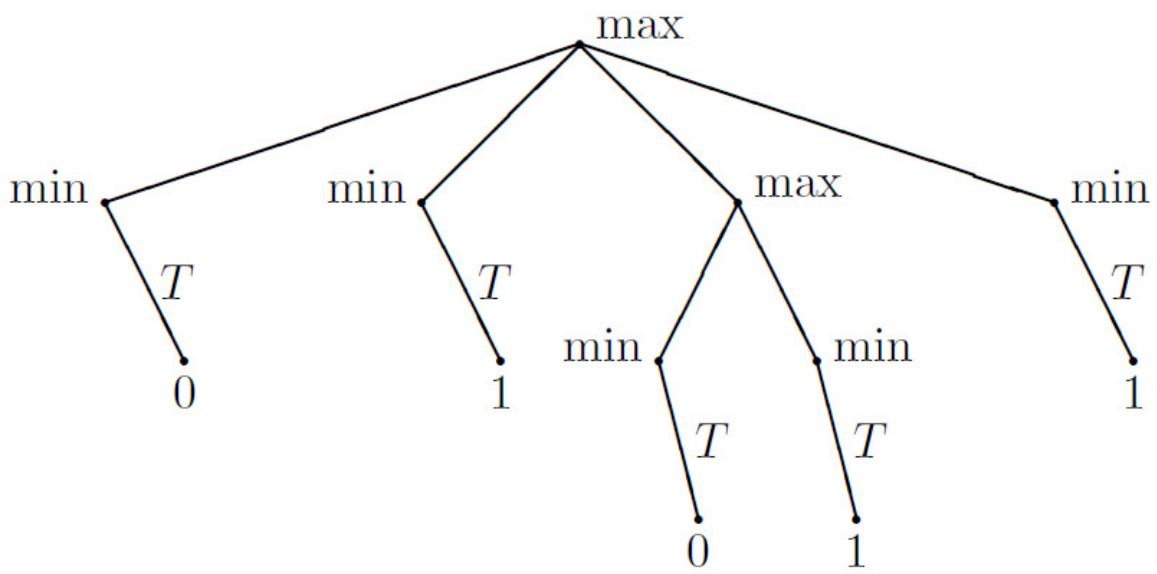
Przykład



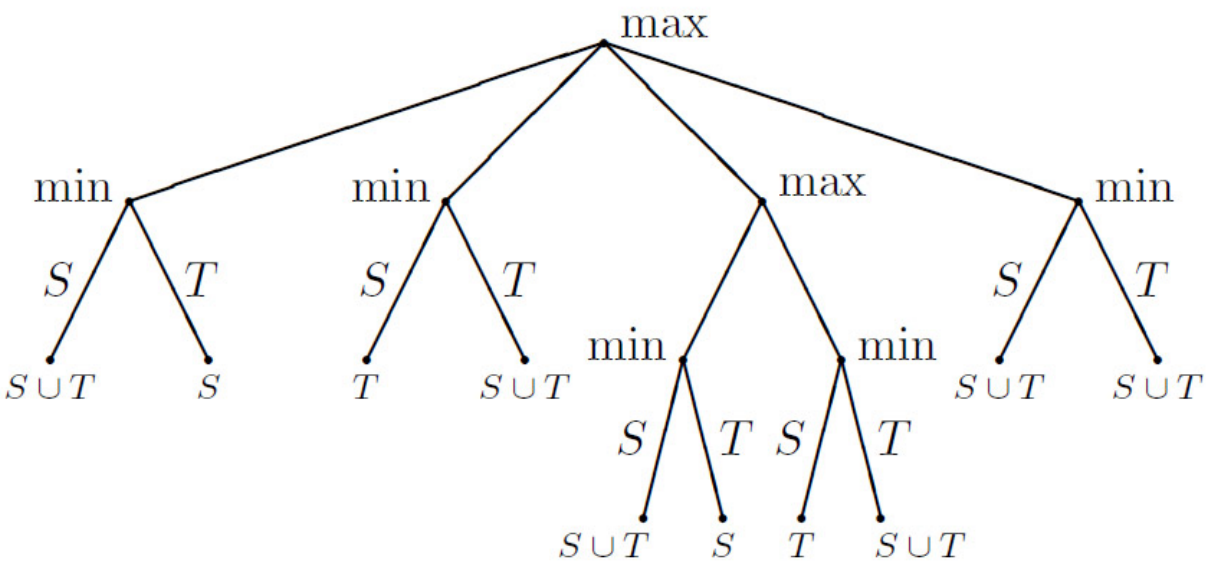
Przykład



Przykład



Przykład



Komplikujemy

Niestety, to nie jest dokładnie to, o co nam chodziło. MAX dalej korzysta z informacji zupełnej, więc rozgrywka wygrywającą w T albo S jest tak samo dobra jak w T lub S.
Ale posłuchajmy jeszcze raz graczy:

- mogę grać na kiery 3-3 albo D♠ u W. To drugie jest lepszą szansą.

To znaczy, że dla danej pozycji nie rozważamy zbioru układów kart dla których wygramy, lecz zbiór tych zbiorów.

Komplikujemy

Z pozycją utożsamiamy zatem funkcję $f : S \rightarrow 2^{2^S}$. Funkcja wyboru dla zawodnika MAX jest dana wzorem:

$$max(\mathcal{F}, \mathcal{G}) = \mathcal{F} \cup \mathcal{G}$$

Dzieje się tak, gdyż gracz MAX może wybrać czy "grać na coś" z $\mathcal{F}$ czy "na coś" z $\mathcal{G}$, a zatem może się zdecydować "na coś" z $\mathcal{F} \cup \mathcal{G}$ Funkcja dla gracza MIN jest natomiast dana wzorem:

$$max(\{F_i\}, \{G_j\}) = \{min(F_i, G_j)\}$$

gdzie $min(F, G)(s) = min(F(s), G(s))$ (tu zakładamy, że MIN gra z informacją zupełną)

Formalnie II

Grą z niezupełną informacją nazywamy $\langle G, V, p_I, s, ev, f_+, f_- \rangle$ gdzie:

- G jest skończonym zbiorem pozycji
- $V = 2^{2^S}$ jest zbiorem wartości gry
- $(p_0, S) = p_I \in G$ jest pozycją początkową
- funkcja s jest zdefiniowana następująco:
  - jeśli gra MAX, to zagrywa jedną z możliwych kart a Z się nie zmienia
  - jeśli gra MIN, to zagrywa jedną z możliwych kart a Z' jest podzbiorem Z, dla którego zagranie tej karty jest możliwe
- $ev : G \rightarrow \{\mathtt{max}, \mathtt{min}\} \cup V$ zwraca wartość pozycji dla pozycji końcowych oraz określa kto gra w pozostałych
- pozycje końcowe, to te kiedy zostaną zagrane wszystkie karty. Wartością gry jest $\{\{s\}, \{S\}\}$ jeśli wygraliśmy oraz $\{\{S\}, \{S - \{s\}\}\}$ gdy przegraliśmy
- $f_+ : \mathcal{P}(V) \rightarrow V$ i $f_- : \mathcal{P}(V) \rightarrow V$ są funkcjami dla gracza

Poprawność II

Załóżmy, że wartością gry G jest pewna rodzina zbiorów $\mathcal{T}$. Wtedy T jest podzbiorem pewnego elementu $\mathcal{T}$ wtw gdy gracz MAX ma strategię wygrywającą dla każdego układu kart $t \in T$ zakładając, że gracz MIN gra z informacją zupełną.

- jak korzeń jest końcowy to OK
- (dla MAX) jeśli MAX może wygrać w T to znaczy, że może przejść do pozycji (p_i, S_i) o wartości $\mathcal{F}_i$, gdzie $T \subseteq F \in \mathcal{F}_i$. Z definicji f_+ mamy zatem, że $T \subseteq F \in \mathcal{F}$, gdzie $\mathcal{F}$ jest wartością (p, S)
- jeśli MAX nie może wygrać, to $T \nsubseteq F \in \mathcal{F}_i$ dla żadnego i a zatem z definicji f_+ mamy, że $T \nsubseteq F \in \mathcal{F}$
- (dla MIN) nieco trudniej ale podobnie: jeśli MAX wygrywa to $T \subseteq F_i \in \mathcal{F}_i$ a zatem $T \subseteq \bigcap F_i \in min(\mathcal{F}_i)$
- jeśli MAX przegrywa to istnieje i takie, że $T \nsubseteq F_j \in \mathcal{F}_i$. A zatem nie istnieje $V \supseteq T$, takie że $V \in min(\{F_j\}, \{U_k\})$

Przykład