MiNI-Player – podsumowanie dotychczasowych badań

Maciej Świechowski
promotor: prof. dr hab. Jacek Mańdziuk

O czym opowiem

- Wprowadzenie – motywacja
- Monte-Carlo Tree Search i UCT
- Strategie gry
- Mechanizm oceny i wyboru strategii
- Mechanizm wyboru ruchu podczas gry
- Środowisko testowe
- Wyniki
- Wnioski, plany i oczekiwania

O czym nie opowiem

- Nie będzie elementarnego wprowadzenia do GGP
- Język reguł: GDL
- Interpreter reguł – część MiNI-Player
- Parallelizacja
- Testowane i odrzucone podejścia

Czym jest MiNI-Player

- Agent GGP; program grający w „dowolne” gry mieszczące się w standardzie GGP i GDL-I.
  - skończone i deterministyczne
- Brał udział w konkursie **GGP Competition 2012**
  - wersja z konkursu jest głównym tematem prezentacji

Trendy – zwycięzcy mistrzostw

2005: Cluneplayer

- uogólnione przeszukiwanie drzewa typu min-max
- funkcja ewaluacyjna: wypłata, kontrola, mobilność

2006: FluxPlayer

- wykrywanie struktur semantycznych w opisie gry
  - plansze, kamienie, liczniki, relacje porządku
- logika rozmyta – stopień spełnienia reguł *goal* i *terminal*
- funkcja ewaluacyjna; zmodyfikowany IDDFS

Trendy – zwycięzcy mistrzostw

2007, 2008, 2012: CadiaPlayer

- Podejście czysto symulacyjne: „Knowledge-Free”
- Symulacje Monte-Carlo i algorytm UCT
- Heurystyka historyczna
- Duży nacisk na statystyczną optymalizację procesu MCTS
- Kontrola symulacji: FAST, MAST, TO-MAST, RAVE i inne
- Z czasem wiele prostych dodatków na zasadzie zgadywania i weryfikacji hipotezy

Trendy – zwycięzcy mistrzostw

2009, 2010: Ary

- Monte-Carlo i UCT
- Podejście podobne do CadiaPlayer; prawdopodobnie mniej rozbudowane
- Nested Monte-Carlo do gier jednoosobowych
- Duży wysiłek badawczy w zakresie technik zrównoleglania

Trendy – zwycięzcy mistrzostw

2011: TurboTurtle

- „Simulation-Based” – wg prezentacji autora o GGP
- brak publikacji – nie wiadomo wiele o tym graczu

Trendy – zwycięzcy mistrzostw

2011: TurboTurtle

- „Simulation-Based” – wg prezentacji autora o GGP
- brak publikacji – nie wiadomo wiele o tym graczu

2013: ??? ☺

„AI systems are dumb because
AI researchers are too clever.”

JACQUES PITRAT (1968)

Motywacja

- Usprawnienie losowych symulacji poprzez wprowadzenie wiedzy
- Próba bezpiecznej aplikacji wiedzy

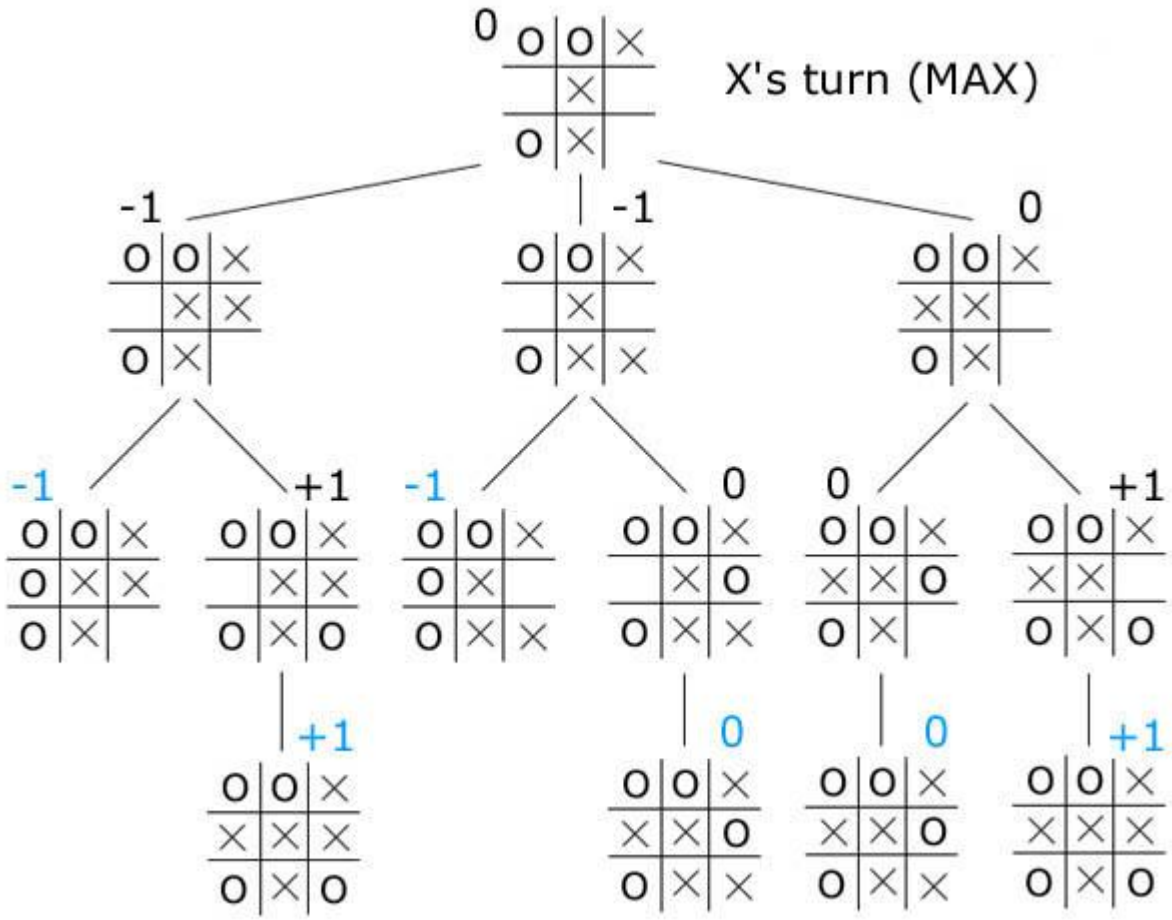
  MCTS vs Min-Max/Alfa-Beta/MTD(f)
- Próba dalszego udoskonalenia podejścia, które uznawane jest za *state-of-the-art* w General Game Playing

MCTS/UCT vs MinMax/αβ/MTD(f)

- Łatwe zrównoleglenie
- Podejście skalowalne
- Można w każdej chwili przerwać
- Symulacje mogą „robić błędy”
- W podstawowej wersji wystarczy logika gry (knowledge-free)
- Potrzebuje silnej i bezbłędnej funkcji oceny stanu
- W przypadku posiadania takiej funkcji, ma przewagę nad MCTS

POJĘCIA BAZOWE

Drzewo gry



[źródło: http://www.ocf.berkeley.edu/~yosenl]

Drzewo gry

Dowolna liczba graczy

- **Wierzchołek w drzewie**: stan w grze
- **Krawędź (A,B)**: wektor akcji każdego z graczy (akcja połączona: *joint move*) powodująca zmianę stanu z A do B

  np. (noop (mark 1 1 x))

Można nie powtarzać tych samych stanów, do których istnieje wiele ścieżek => graf skierowany

Monte-Carlo Tree Search (MCTS)

Wykonujemy losowe symulacje i przechowujemy statystyki w węzłach:

- sumaryczny wynik gry każdego z graczy
- liczbę odwiedzin w węźle
- średni wynik – wyliczany na podstawie powyższych dwóch

Podejście typu knowledge-free.

Wystarczy zaimplementować mechanikę gry

Monte-Carlo Tree Search (MCTS)

- **Selekcja**: w każdym węźle rozpoczynając od wierzchołka wybieramy najlepszego potomka do kontynuacji przeszukiwania
- **Rozwinięcie**: gdy jesteśmy w liściu to tworzymy nowy węzeł
- **Symulacja**: gramy losowo aż do osiągnięcia stanu terminalnego
- **Propagacja wsteczna**: uaktualniamy statystyki – średni wynik oraz liczbę odwiedzin w każdym węźle na ścieżce, którą wybraliśmy

Monte-Carlo Tree Search (MCTS)

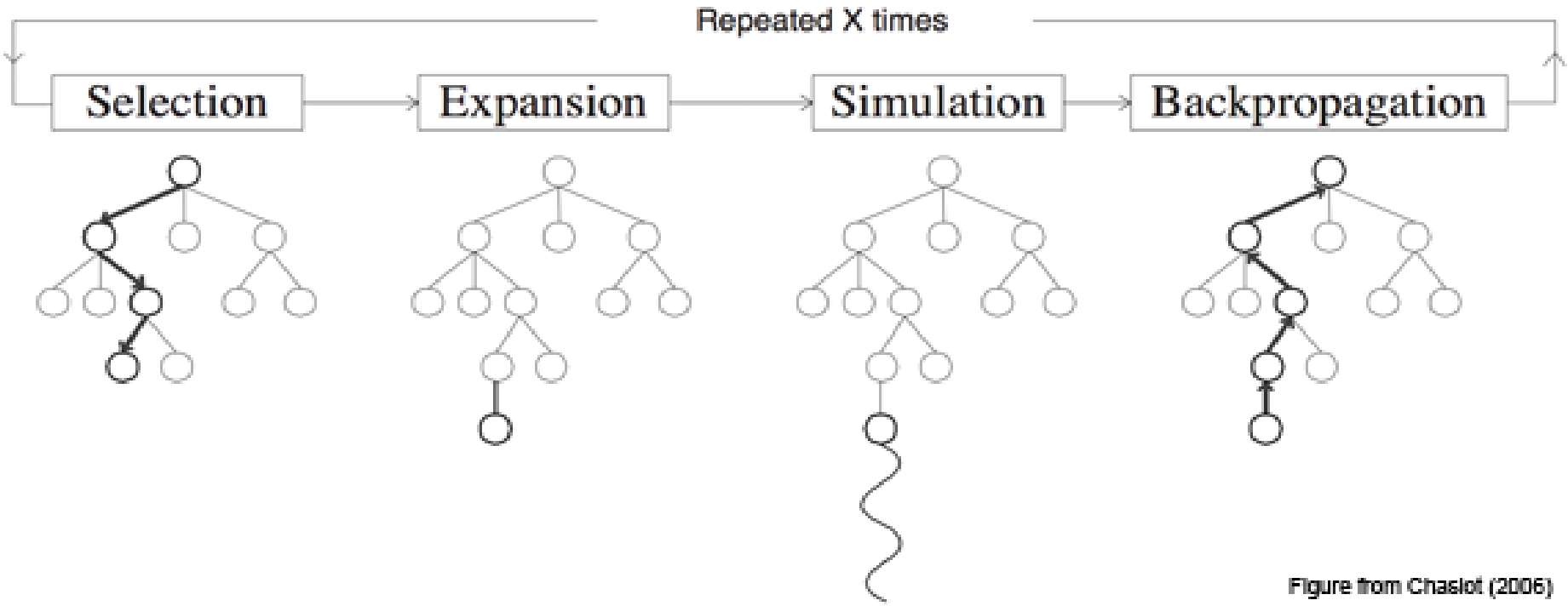


Figure from Chaslot (2006)

UCT *(Upper Confidence Bounds Applied For Trees)*

- Modyfikacja algorytmu **UCB**

  [k-armed/multiarmed bandit problem]

Bandyta: Stała, nieznana dystrybuanta wypłat

Cel: maksymalizacja zysku

UCT *(Upper Confidence Bounds Applied For Trees)*

$$a^* = \arg\max_{a \in A(s)} \left\{ Q(s,a) + C \sqrt{\frac{\ln N(s)}{N(s,a)}} \right\}$$

Dla danej roli (gracza):

a – akcja

s – bieżący stan

Q(s,a) – uśredniony osiągnięty wynik gry przy wykonaniu akcji **a** w stanie **s**

N(s) – liczba dotychczasowych odwiedzin stanu **s**

N(s,a) – liczba dotychczasowych wyborów akcji **a** w stanie **s**

UCT *(Upper Confidence Bounds Applied For Trees)*

Zasada balansu między **eksploracją** nowych ścieżek a **wykorzystaniem** najlepszych do tej pory.

klasyczny problem **exploration** vs **exploitation**

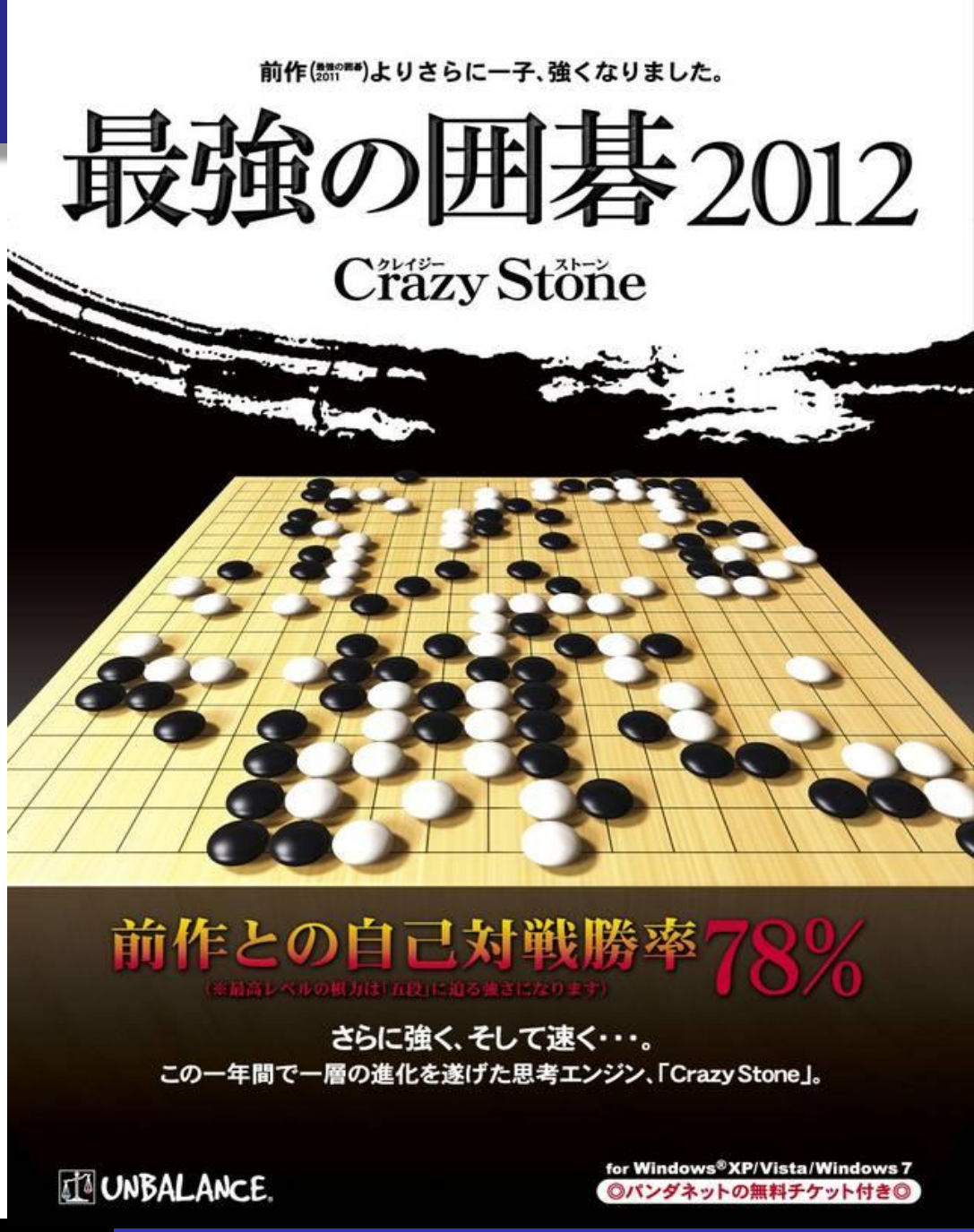
Celem jest ukierunkowanie symulacji na najbardziej obiecujące fragmenty w drzewie.

[L. Kocsis and C. Szepesvari, 2006]

CrazyStone

2006

Komercyjne zastosowanie UCT



Zastosowanie UCT w MiNI-Player

- Optymalizacja fazy **selekcji**
- Dla każdego z graczy – stosowane osobne UCT

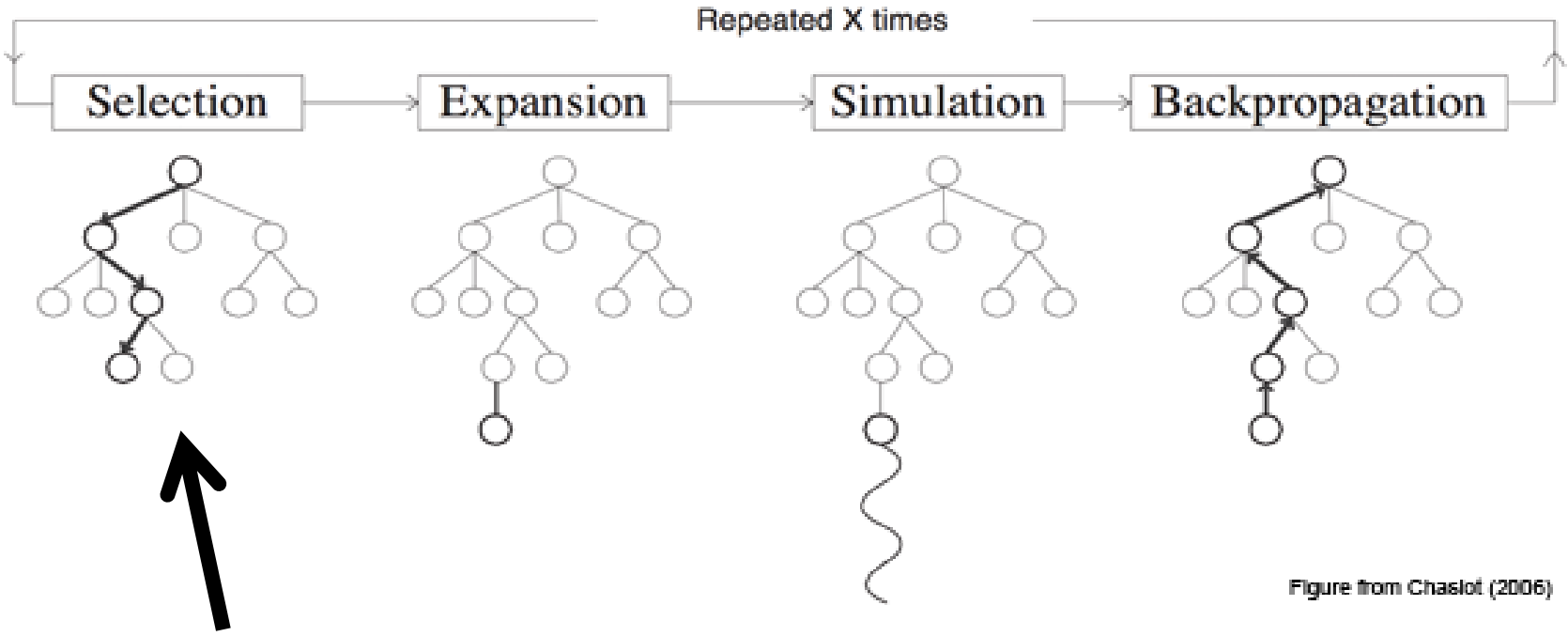


Figure from Chaslot (2006)

Warunki stawiane symulacji w Go

Ostatnie badania w Go, grze której rozwiązania zainspirowały twórców najsilniejszych programów GGP wskazują, że symulacje MCTS powinny:

- **Wybierać silne akcje, które zagrałby gracz**
- **Być różnorodne**

STRATEGIE SYMULACJI

Zastosowanie strategii w MiNI-Player

- Optymalizacja fazy **symulacji**

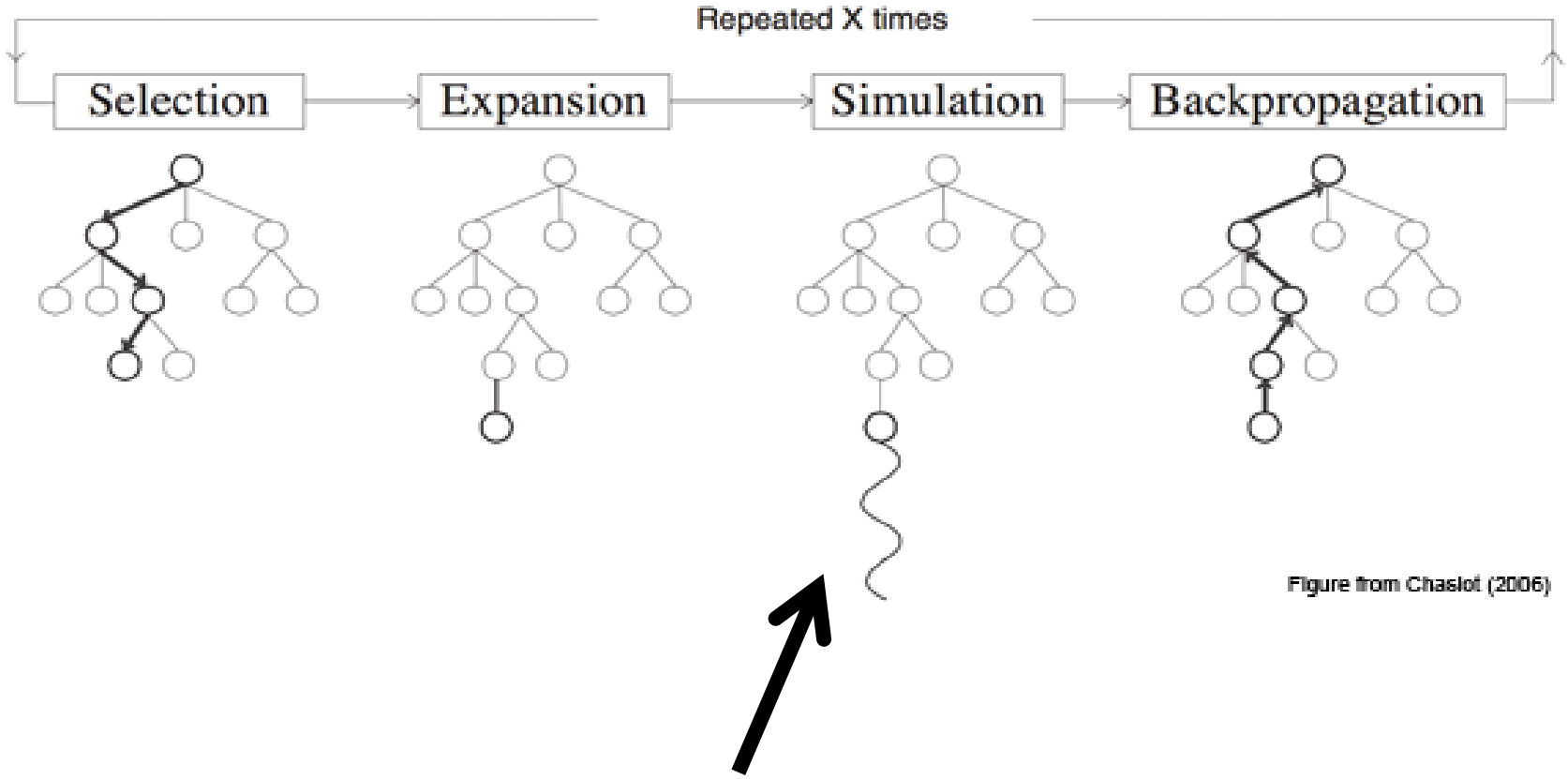


Figure from Chaslot (2006)

Schemat symulacji w GGP

dopóki(krok < MAX_STEP)

1. Wyznacz legalne akcje [**legal**]
2. Dla każdej roli i=0…N [**does**]

   move[i] = **wybierz akcję**

   wykonaj(move[i])
3. Wyznacz, a następnie uaktualnij stan gry [**next**]
4. Sprawdź czy stan jest terminalny: [**terminal**]

   TAK: pobierz wyniki i zakończ symulację [**goal**]

   NIE: wróć do punktu 1.

Dostępne strategie

W turniejowej wersji MiNI-Player:

- Random
- AGE (Approximate Goal Evaluation)
- HH (History Heuristic)
- M (Mobility)
- E (Exploration)
- SSC (Statistical Symbol Counting)

Kandydaci do przetestowania:

- M2 (Mobility 2-ply) i inne wersje 2-ply
- Sc (Score)

Random

- Strategia losowa: wszystkie akcje tak samo prawdopodobne

- Niezwykle ważna, bo:
  - Zwiększa różnorodność przeszukiwania – nie trzyma się konkretnej ścieżki
  - Wydajna obliczeniowo, co pozytywnie wpływa na ilość wykonywanych symulacji

(marginalnie szybsze byłoby tylko zawsze wybieranie akcji o stałym indeksie)

AGE

- Przybliżony stopień spełnienia warunku ***goal***

  wybierany jest zbiór reguł *goal* z najwyższą wypłatą dla każdego gracza

- Podobna idea do FluxPlayer, lecz inne wykonanie.

AGE – pojęcie wstępne

- Relacja: **knight_move** (zbiór wszystkich faktów i reguł)
- Reguły:

```
(<= (knight_move ?piece ?u ?v ?x ?y ?owner)
   (piece_owner_type ?piece ?owner knight)
   (adjacent_two ?v ?y)
   (adjacent ?u ?x))

(<= (knight_move ?piece ?u ?v ?x ?y ?owner)
   (piece_owner_type ?piece ?owner knight)
   (adjacent_two ?u ?x)
   (adjacent ?v ?y))
```

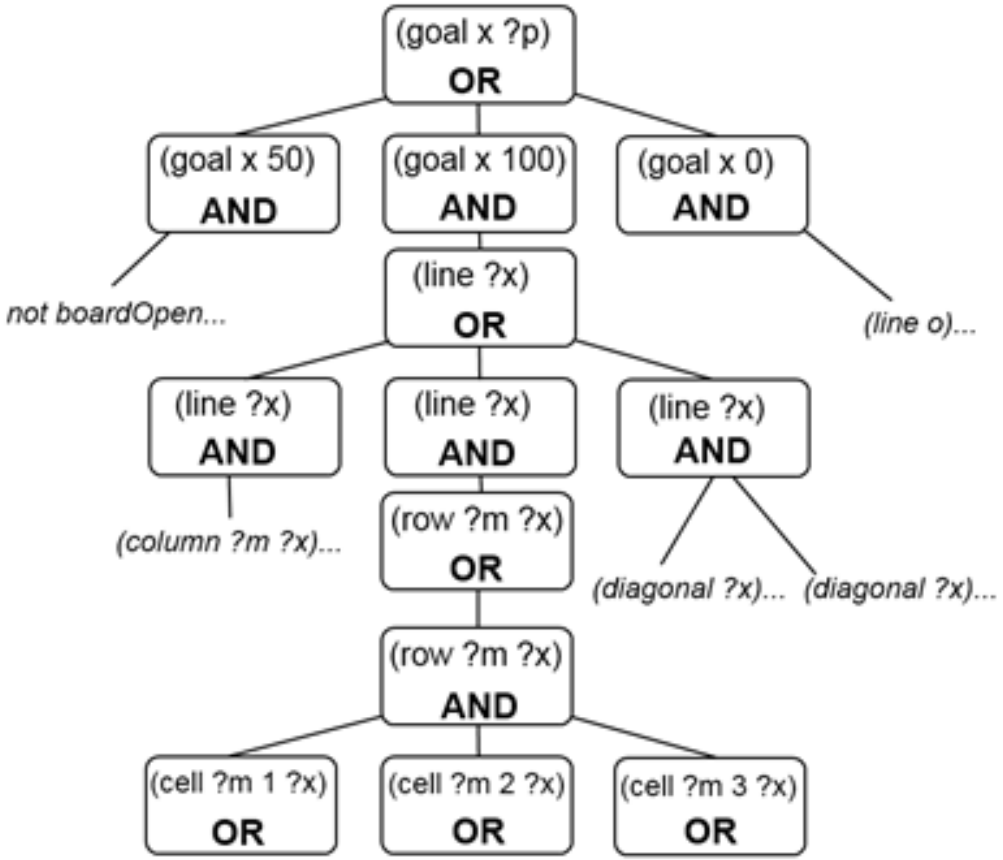
- Warunek ⇔ Zapytanie ⇔ Wywołanie relacji:

  (adjacent_two ?u ?x)

  (knight_move ?piece c 4 ?x ?y white)

AGE

Reprezentacja każdej reguły jako drzewa OR-AND



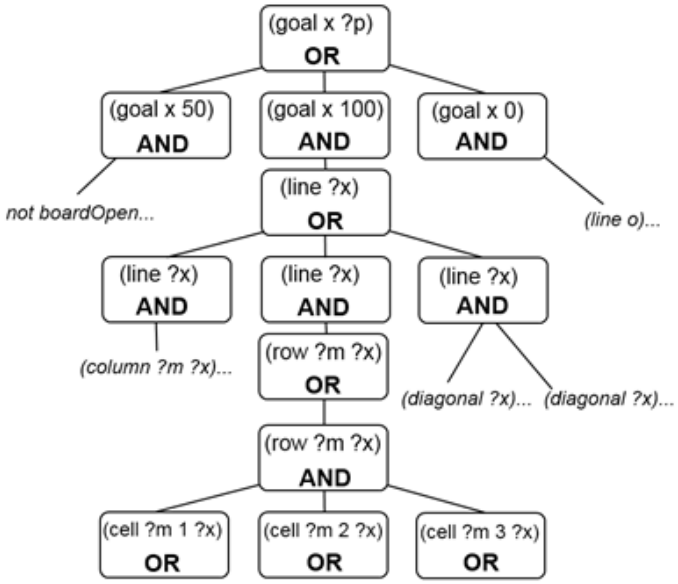
AGE

Reprezentacja każdej reguły jako drzewa OR-AND

- **OR**: reprezentuje kontener na wszystkie zdefiniowane reguły dla danej relacji np. goal, legal, cell

(czyli de facto zapytanie o fakty danego typu – bo pytając nie specyfikujemy konkretnej instancji reguły)

- **AND**: reprezentuje konkretną instancję reguły



AGE

Metoda obliczania AGE jest zdefiniowana rekurencyjnie na drzewie.

- rozpoczyna się w korzeniu
- obliczane i propagowane wstecz są 2 wartości: **AgVal** i **TbVal** (**A**pproximate**G**oal**Val**ue i **T**ie**B**reaker**Val**ue)

Możliwa dzięki własnej implementacji interpretera reguł.

AGE – węzły typu AND

i=1…N – węzły OR będące dziećmi bieżącego węzła AND

$$AgVal = \frac{1}{N}\sum_{i=1}^{N} AgVal_i$$

$$TbVal = \frac{1}{N}\sum_{i=1}^{N} TbVal_i$$

Jeżeli warunek reprezentowany przez OR posiada negację, to wartości wchodzą do sumy jako $1 - AgVal_i$ oraz $1 - TbVal_i$

AGE – węzły typu OR

1. Liść

$AgVal = 1$, jeśli reguła jest spełniona.

$AgVal = 0$, jeśli reguła nie zachodzi

Innymi słowy, pytamy czy zachodzi co najmniej jedna realizacja zmiennych dla zapytania reprezentowanego przez węzeł.

2. Węzeł wewnętrzny:

$$AgVal = \max_{i=1\ldots N} AgVal_i$$

$$TbVal = \frac{1}{N}\sum_{i=1}^{N} TbVal_i$$

Heurystyka Historyczna

Uniwersalna metoda stosowana w AI w grach od 1989 [Schaeffer]

Ideą jest, by przenieść informację uzyskaną z wykonanych akcji wcześniej w procesie symulacji do nowej sytuacji, gdy akcja jest znów możliwa do wykonania.

W GGP większość programów stosuje HH, aby zwiększyć prawdopodobieństwo wybrania akcji dobrych niezależnie od stanu podczas symulacji Monte-Carlo.

Heurystyka Historyczna w MiNI-Player

- Globalny słownik akcji dla każdego z graczy
- Akcje wybierane podczas symulacji pozostałymi strategiami są zapamiętywane
- Po zakończeniu symulacji uaktualniane są średnie wyniki

- Podczas symulacji z użyciem HH w każdym kroku z prawdopodobieństwem ε wybierany jest najlepiej oceniony ruch z aktualnie legalnych

  (w przeciwnym przypadku lub gdy takiego nie ma – losowy)

Mobility

Liczba dostępnych akcji względem pozostałych graczy

- zwykle im więcej tym lepiej

  (więcej możliwości, więcej figur w grach planszowych)

- Niech M_i oznacza liczbę akcji dostępnych dla i-tego gracza
- Niech i=0 oznacza rolę przypisaną MiNI-Playerowi

Wybieramy stan, w którym maximum osiąga wyrażenie:

$$M_0 + \sum_{i=1}^{N-1} (M_0 - M_i)$$

Exploration

Idea tej strategii polega na odwiedzaniu nowych stanów, które w największym stopniu różnią się od ostatnio odwiedzanych.

Exploration

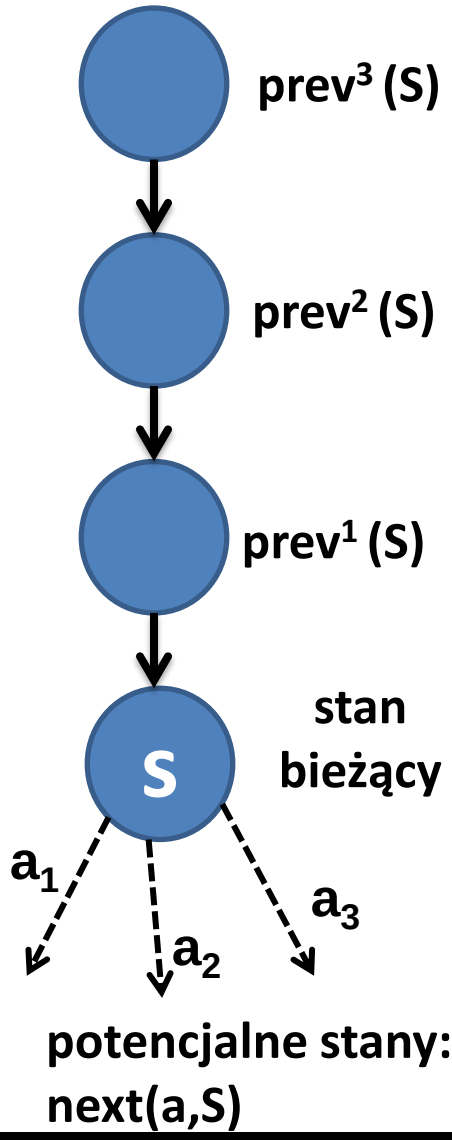
Idea tej strategii polega na odwiedzaniu nowych stanów, które w największym stopniu różnią się od ostatnio odwiedzanych.

Różnica między stanem B i A:

diff(B,A) = B – A $\quad |\{f : f \in B \wedge f \notin A\}|$

- Liczność zbioru faktów (termów GDL), które należą do B oraz nie należą do A

Exploration



Dla każdej legalnej akcji a_i w bieżącym stanie S wyznacz stan następny $S'_i = next(a_i,S)$

Dla każdego i wyznacz najbardziej podobny stan z ostatnich N:

$$\min_i d_i = \min_{j=1..N} diff[S'_i, prev^j(S'_i)]$$

Wybierz akcję, której najbardziej podobny stan jest od niej maksymalnie różny:

$$a_i := arg \max_i \left(\min_i d_i \right)$$

Statistical Symbol Counting

Idea zaczerpnięta z wcześniej testowanego podejścia do budowania funkcji ewaluacyjnej.

- ***wzięty podzbiór (prostsza część)***

Strategia ma dwa fazy:

- **uczącą**: w czasie START CLOCK
- **grającą**: po zakończeniu fazy uczącej

Faza ucząca – wykorzystuje symulacje przeprowadzane przez pozostałe strategie, aby zbierać informacje

Statistical Symbol Counting

Podczas symulacji uczących:

Dla każdej **relacji** zdefiniowanej w GDL obecnej w stanie gry:

- wyznaczana jest liczność faktów (realizacji)

Dla każdej pary: **indeks** na liście argumentów realizacji (nr kolumny) oraz **symbol**:

- wyznaczana jest liczba symboli

Wszystkie liczności uśredniane są dla symulacji.

(vs liczność z końcówki, vs średnia ważona długością symulacji)

Statistical Symbol Counting

[cell 1 1 x]
[cell 1 2 b]
[cell 1 3 x]
[cell 2 1 b]
[cell 2 2 o]
[cell 2 3 b]
[cell 3 1 b]
[cell 3 2 b]
[cell 3 3 b]

Quantity(”cell”) = 9

Quantity(”cell”, 1, ”1”) = 3
Quantity(”cell”, 1, ”2”) = 3
Quantity(”cell”, 1, ”3”) = 3
Quantity(”cell”, 2, ”1”) = 3
Quantity(”cell”, 2, ”2”) = 3
Quantity(”cell”, 2, ”3”) = 3
Quantity(”cell”, 3, ”x”) = 2
Quantity(”cell”, 3, ”o”) = 1
Quantity(”cell”, 3, ”b”) = 6

Optymalizacje odcinające po pewnym momencie niektóre kolumny.

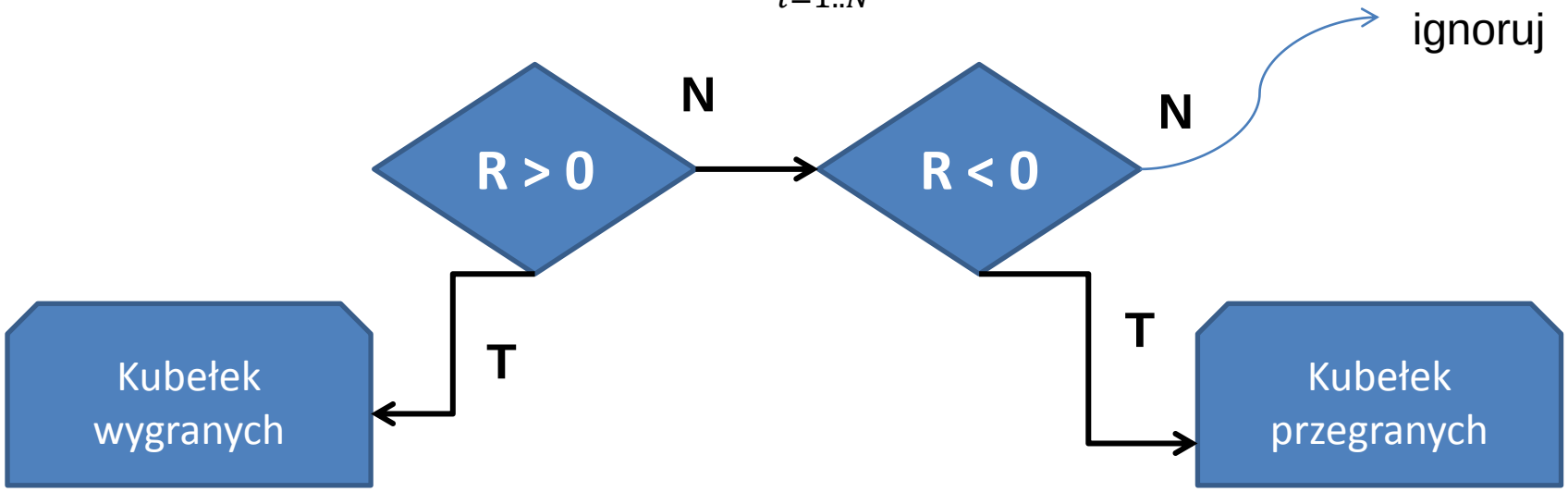
Statistical Symbol Counting

Średnia wartość danej cechy może zostać dodana do jednego z dwóch kubełków na podstawie wyniku zakończonej symulacji:

i = 0 – indeks roli przypisanej do MiNI-Player

N – liczba graczy

$$\mathrm{R} = goal_0 - \max_{i=1..N}(goal_i)$$



Statistical Symbol Counting

Średnie wartości są dalej uśredniane oddzielnie dla całej przestrzeni kubełków:

- Gier wygranych
- Gier przegranych

Cesze przypisywana jest waga:

$w = AVG_{wins} - AVG_{losses}$

Odrzucane są cechy z wagami |w|< 0.0001

Tworzona jest funkcja oceny stanu jako kombinacja liniowa:

liczność cechy * waga cechy

Dobór parametrów

Nawet wybrana strategia do przeprowadzenia symulacji ma pewne prawdopodobieństwo P wykonania akcji zgodnie ze swoją formułą.

- aby zapewnić różnorodność przebiegów

P	AGE Checkers	SSC Chess	M Othello	E Farming Quandries	HH Connect-4
0.20	59.17	50.00	54.33	54.83	54.67
0.40	66.00	51.00	55.50	56.50	55.83
0.50	65.67	51.33	68.17	58.17	59.33
0.60	70.67	51.00	73.83	59.50	59.00
0.65	79.17	50.83	73.67	60.67	57.33
0.70	76.33	51.33	81.33	61.17	**66.67**
0.75	**79.33**	49.50	83.67	61.67	62.92
0.80	75.83	53.17	81.17	**62.17**	59.17
0.85	72.67	54.17	**85.17**	58.33	54.83
0.90	69.33	54.64	79.17	42.50	50.67
1.00	55.67	**55.00**	68.17	34.83	44.17
95% Conf.	±4.46	±9.07	±3.90	±5.10	±5.13

Potencjalne strategie

- Przeszukiwanie na 2 kroki do przodu
- Wykrycie liczników – faktów, które zawierają w danym momencie jedną wartość dla każdego z graczy

  (maksymalizacja/minimalizacja tych wartości)

- Wykrywanie planszy?
- N-grams? Szeregi akcji do wykonania w przypadku gier o niewielkiej interakcji między graczami?
- Obliczanie odległości między faktami – minimalizacja/maksymalizacja/koncentracja

sugestie? ☺

Mechanizm wyboru strategii

Testowane rozwiązania:

1. Proporcjonalnie do średniego wyniku
2. Proporcjonalnie do kwadratu średniego wyniku
3. Statyczna alokacja na N symulacji bazująca na liście rankingowej strategii
4. Algorytm UCB

1. Proporcjonalnie do Q

- Weźmy dwie strategie S_1 oraz S_2 o ocenach

  $Q_1 = 0.6$, $Q_2 = 0.5$

Przydzielając ilości symulacji N dla strategii proporcjonalnie do Q, otrzymujemy:

$N_1 = 120$

$N_2 = 100$

po 220 symulacjach!

2. Proporcjonalnie do Q^2

Planujemy kolejkę strategii przydzielanych do symulacji:

1. Znajdujemy **d** = **min(Q_i)** – „najsłabszą” strategię
2. Obliczamy znormalizowane wartości średnie dla każdej strategii:

$$AVG_i = \frac{Q_i}{d}$$

3. Przydzielamy i-tej strategii ilość symulacji N_i:

$$N_i = \lfloor R_i \rfloor, \quad R_i = (AVG_i + OF_i)^2$$

4. Obliczamy przepełnienie *OF* w bieżącym kroku:

$$OF_i = R_i - \lfloor R_i \rfloor$$

2. Proporcjonalnie do Q^2

Planujemy $\sum N_i$ następnych symulacji

- równomiernie przeplatając
- startując od najlepiej ocenionych strategii

Przykład:

$N_1 = 4$, $N_2 = 3$, $N_3 = 1$

Wynik => plan symulacyjny:

[S1, S2, S1, S2, S3, S1, S2, S1]

3. Statyczna alokacja

- Budujemy listę rankingową strategii sortując po średnim wyniku $\mathbf{Q_i}$
- Przydzielamy z góry ustaloną liczbę symulacji w zależności od zajmowanego miejsca na liście

Przykład (dla MiNI-Player):

Ranking: [S3, S2, S6, S1, S4, S5]

Alokacja: [22, 11, 7, 4, 2, 1]

4. UCB

- Tu nie planujemy z góry symulacji, tylko decyzje podejmujemy w każdym kroku bazując na formule:

$$S^* = arg \max_{i=1...6} \left\{ Q_i + C \sqrt{\frac{\ln n}{T(S_i, n)}} \right\}$$

S^* - wybrana strategia

n - liczba wszystkich symulacji od początku

$T(S_i,n)$ – liczba wyborów strategii **S_i** wśród **n** symulacji

C – współczynnik eksploracji; **C** = 5 (wersja na mistrzostwa)

Porównanie mechanizmów selekcji

- AVG (Q^2)
  - zbyt słabe wyeksponowanie najlepszych strategii
- UCB
  - sprawdzona metoda, uzasadniona statystycznie
  - ryzyko zbytniej dominacji jednej strategii
- Static
  - w praktyce okazuje się bardzo dobra
  - nieoptymalna w ekstremalnych przypadkach: istnienia ewidentnie złej lub ewidentnie jedynej dobrej strategii

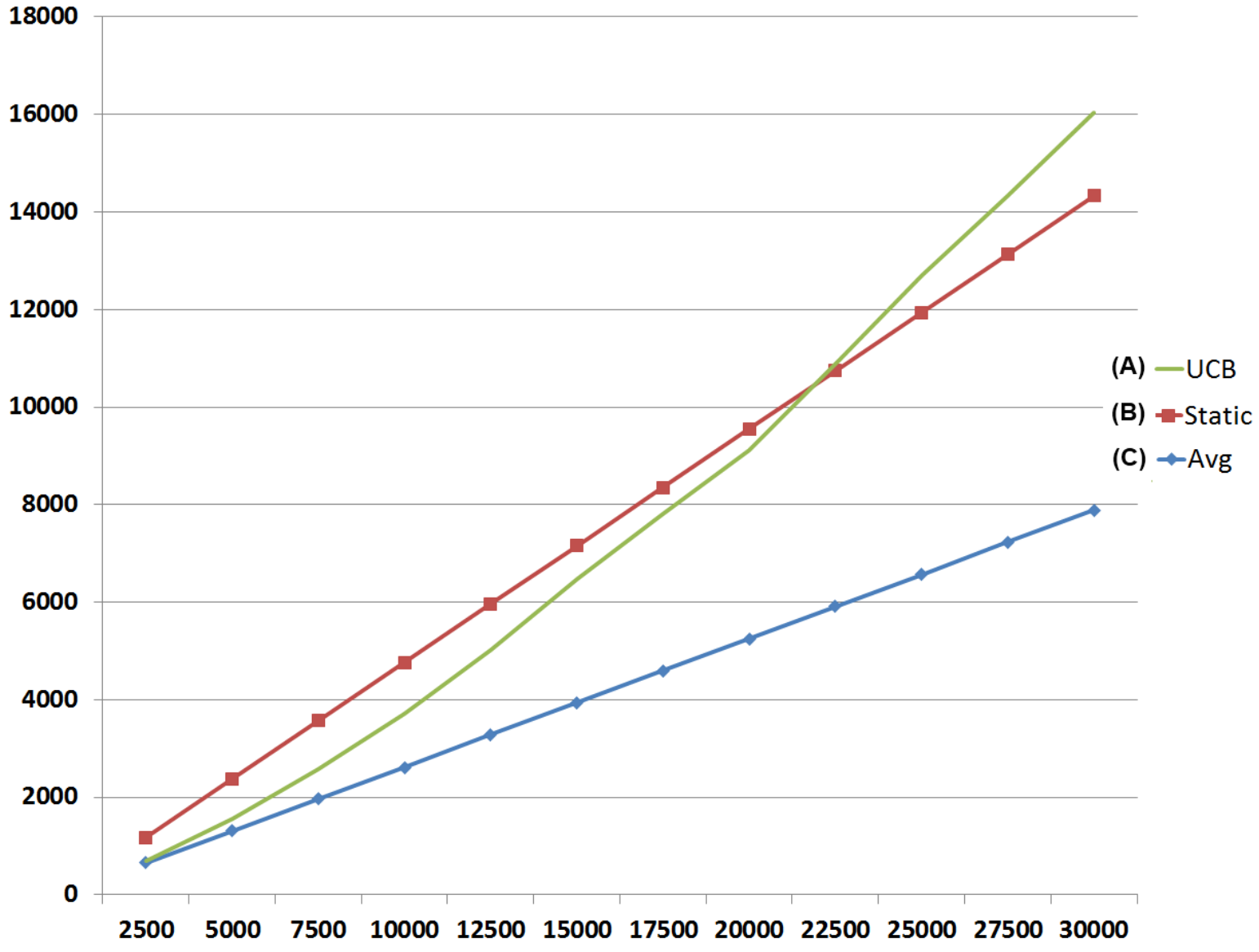
Porównanie mechanizmów selekcji

Koszt obliczeniowy:

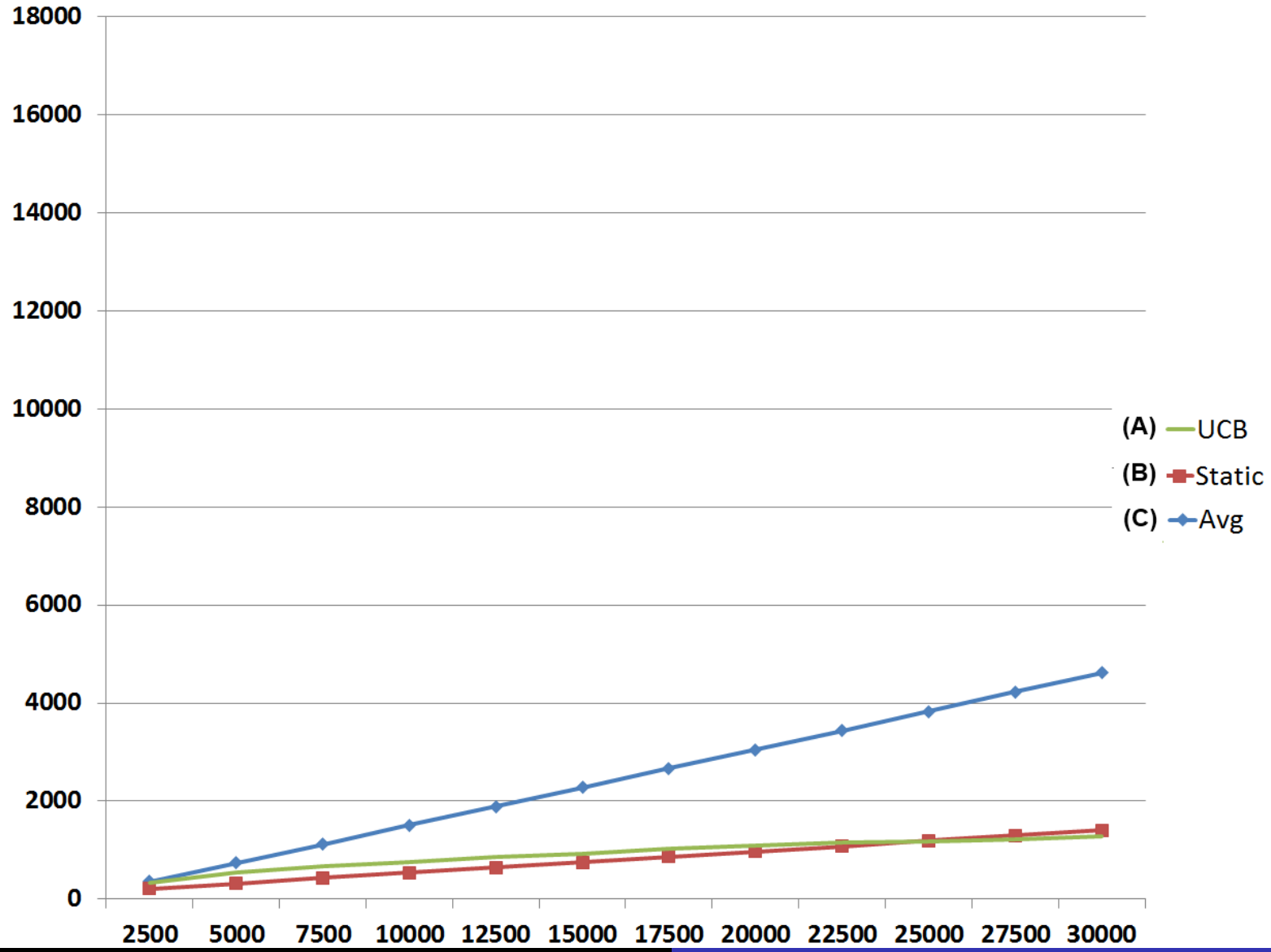
Static > AVG (Q^2) > UCB

w praktyce pomijalny w stosunku do czasów symulacji

Alokacja symulacji dla najlepszej strategii



Alokacja symulacji dla najgorszej strategii



Mechanizm wyboru akcji

Akcja – składowa wektora w krawędzi wychodzącej z aktualnego korzenia

Wersja klasyczna przy korzystaniu z UCT:

- Wybierz akcję prowadzącą do potomka o **najlepszym średnim wyniku Q**
- Wybierz akcję najczęściej symulowaną

(w wyjątkowych sytuacjach oba przypadki nie są tożsame)

Mechanizm wyboru akcji

- MiNI-Player korzysta ze zmodyfikowanej formuły **EQ**, która wykorzystuje dane z 2 poziomów w drzewie
- Wybrany zostaje potomek **Root.Child[i]** o największej wartości **EQ**

Obliczanie EQ

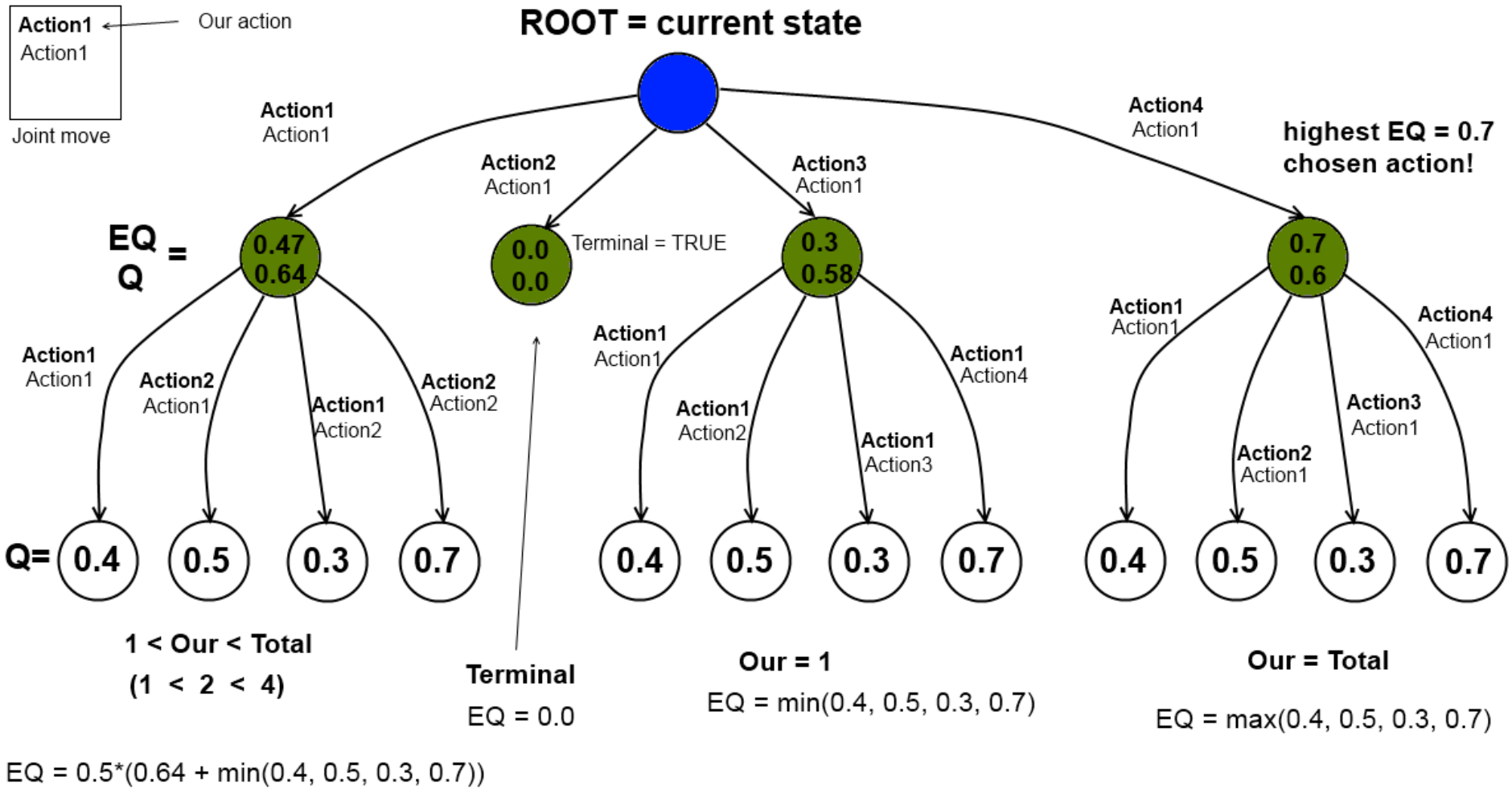
Our – liczba unikalnych akcji naszego gracza

Total – liczba wszystkich dostępnych akcji łączonych (joint moves)

Iterujemy po węzłach wychodzących z korzenia:

Warunek w węźle	EQ
Stan terminalny = TRUE	$\mathrm{EQ} = node.Q \qquad (1.01 * node.Q)$
Our = 1	$\mathrm{EQ} = \min_{i=1\ldots N}(node.Child[i].Q)$
1 < Our < Total	$\mathrm{EQ} = \frac{\left(\min_{\mathrm{i=1\ldots N}}(\mathrm{node.Child[i].Q}) + \mathrm{node.Q}\right)}{2}$
Our = Total	$\mathrm{EQ} = \max_{i=1\ldots N}(node.Child[i].Q)$

Mechanizm wyboru akcji



EQ = 0.5*(0.64 + min(0.4, 0.5, 0.3, 0.7))

Tabele transpozycji

- Stosowane w grach, aby nie duplikować w drzewie powtarzających się stanów
- Procedura znalezienia identycznego stanu (porównywania stanów) musi być bardzo wydajna
  - sprawdzanie po każdym napisie GDL nie wchodzi w grę, zwłaszcza, że mogą być w dowolnej kolejności
  - Pesymistyczna złożoność: N*R*C

    N: liczba węzłów w drzewie

    R: liczba relacji składających się na stan

    C: średnia liczba faktów występującej dla danej relacji

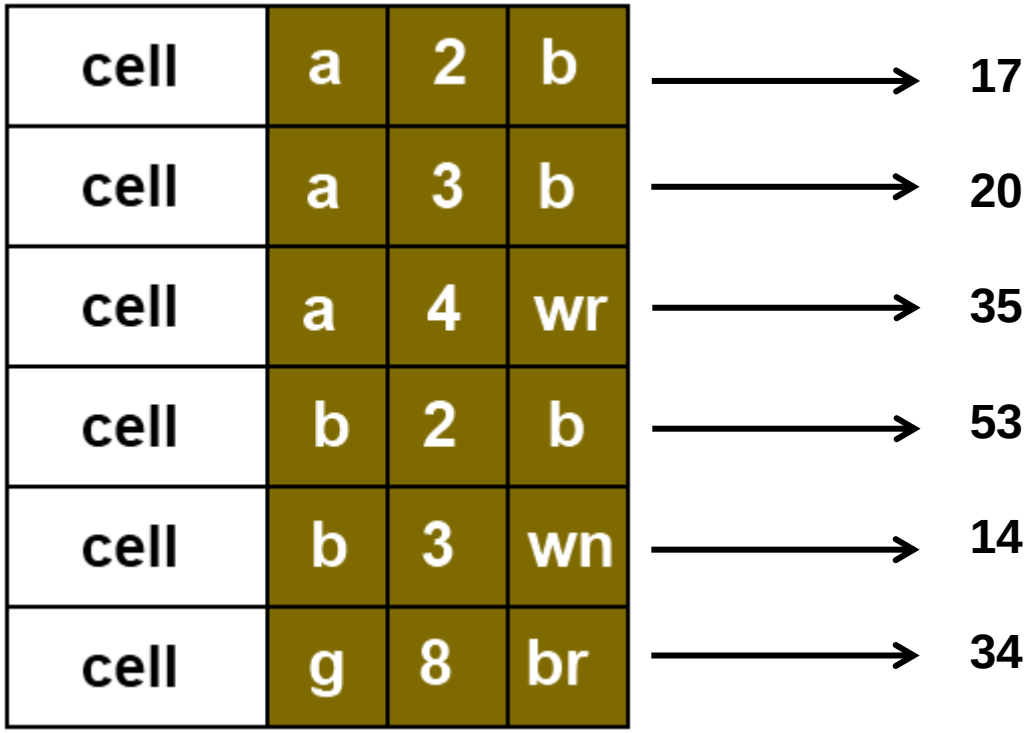
Tabele transpozycji

- W grach znanych z góry stosuje się bardzo wydajne reprezentacje stanów np. Zobrist Hashing/Keys

- Przede wszystkim znana jest natura gier i wymiary poszczególnych elementów

Tabele transpozycji w MiNI-Player

- Funkcja hashującą dla realizacji GDL efektywną w obrębie swojej relacji:



- Funkcja hashująca dla całej relacji – suma + zoptymalizowana metoda EqualTo do radzenia sobie z kolizjami

Tabele transpozycji w MiNI-Player

Klasyfikacja stanów, ze względów na pewne cechy celem przyspieszenia

- Sumaryczna liczba legalnych akcji
- Sumaryczna ilość realizacji
- Ilość niepustych relacji

  (w sensie liczności zbioru realizacji)
- Ilość realizacji dla każdej z relacji

  (relacje znane są apriori bezpośrednio z reguł)

Tabele transpozycji w MiNI-Player

Nie ma modyfikacji struktury drzewa.

- Przy dodaniu nowego węzła sprawdzane jest istnienie węzłów transponowanych
- W fazie propagacji wyniku – wynik jest tak samo propagowany na węzły transponowane

Wciąż można przechodzić w drzewie wieloma ścieżkami do tego samego stanu, ale podbicie liczby odwiedzin w połączeniu z algorytmem UCT minimalizuje wynikające z tego problemy.

WYNIKI EKSPERYMENTALNE

Gry z mistrzostw 2012

- Connect-4
- Cephalopod Micro
- Free for All 2P
- Pentago
- 9 Board Tic-Tac-Toe
- Connect-4 Suicide
- Checkers
- Farming Quandries
- Pilgrimage

Wyniki vs CadiaPlayer

5-4 w grach

Game	Clock [s]		MINI vs. Cadia	GGP 2012 Result
Connect4	40	15	41.67 [5.35]	Loss (error)
Cephalopod Micro	60	20	40.00 [5.84]	Loss
Free for all 2P	45	15	63.33 [5.14]	Win
Pentago	45	15	29.33 [5.43]	Loss
9 Board Tic-Tac-Toe	45	15	70.67 [5.16]	Win
Connect4 Suicide	45	15	53.33 [5.26]	Win
Checkers	60	20	54.33 [5.88]	Win
Farming Quandries	90	30	68.33 [4.21]	Loss
Piligrimage	90	30	42.67 [4.32]	Loss
Average			**51.40** [5.18]	

Wyniki gracza bez strategii i EQ

Gracz na bazie MINI-Player, tylko z heurystyką historyczną (bez innych strategii symulacji)

Game	MINI vs. MINI-C	Cadia vs. MINI-C
Connect4	61.33 [5.69]	59.67 [5.54]
Cephalopod Micro	59.33 [5.86]	73.33 [5.27]
Free for all 2P	75.33 [4.86]	65.67 [5.11]
Pentago	40.00 [5.84]	55.33 [5.93]
9 Board Tic-Tac-Toe	66.30 [5.42]	50.00 [5.76]
Connect4 Suicide	51.33 [5.72]	43.33 [5.50]
Checkers	79.33 [4.83]	69.33 [5.32]
Farming Quandries	66.67 [5.36]	59.67 [4.90]
Piligrimage	39.33 [5.40]	62.83 [5.39]
Average	**59.88** [5.44]	**59.91** [5.42]

Wyniki dla gier jednoosobowych

2-1-2 ze średnią na korzyść MiNI-Player

Game	Clock [s]		MINI-Player	CadiaPlayer
Hanoi	30	20	80.70 [0.55]	99.00 [0.52]
Knight Tour	30	20	100.00 [0.00]	100.00 [0.00]
8-Puzzle	30	20	64.80 [4.82]	3.60 [2.10]
Lights Out	30	20	18.50 [4.63]	0.00 [0.00]
Rubik's Cube	30	20	0.00 [0.00]	13.25 [1.17]
Average			**52.80** [2.00]	**43.17** [0.76]

Siła poszczególnych usprawnień

Game	MINI-EQ vs. Cadia	MINI-STR vs. Cadia
Connect4	54.84 [5.85]	40.00 [5.42]
Cephalopod Micro	36.67 [5.75]	40.00 [5.84]
Free for all 2P	39.81 [5.31]	59.56 [5.39]
Pentago	45.93 [5.94]	31.33 [5.53]
9 Board Tic-Tac-Toe	49.33 [5.72]	66.67 [5.41]
Connect4 Suicide	60.67 [5.40]	51.33 [5.33]
Checkers	24.00 [5.05]	53.67 [5.93]
Farming Quandries	39.33 [4.60]	64.67 [4.54]
Piligrimage	35.67 [4.09]	40.33 [4.55]
Average	**42.91** [5.30]	**49.73** [5.33]

Podsumowanie usprawnień

Game	MINI-C	MINI-EQ	MINI-STR	MINI
Connect4	1.00	1.36	0.99	1.03
Cephalopod Micro	1.00	1.38	1.50	1.50
Free for all 2P	1.00	1.16	1.73	1.84
Pentago	1.00	1.03	0.70	0.66
9 Board Tic-Tac-Toe	1.00	0.99	1.33	1.41
Connect4 Suicide	1.00	1.07	0.91	0.92
Checkers	1.00	0.78	1.75	1.77
Farming Quandries	1.00	0.98	1.60	1.69
Piligrimage	1.00	0.96	1.09	1.15
Average	**1.00**	**1.07**	**1.24**	**1.28**

Ocena strategii dla poszczególnych gier

Game	R	AGE	M	HH	E	SSC
Connect4	54.6	78.1	55.00	**80.6**	58.3	54.0
Cephalopod Micro	49.1	34.9	44.3	**57.9**	34.8	42.6
Free for all 2P	52.4	68.0	58.5	76.2	**94.5**	84.5
Pentago	48.9	**75.7**	43.1	47.0	45.4	45.7
9 Board Tic-Tac-Toe	51.3	**71.8**	49.7	32.0	48.5	46.6
Connect4 Suicide	54.1	51.4	49.0	**59.2**	45.0	49.6
Checkers	48.8	77.7	64.1	54.7	75.0	**78.5**
Farming Quandries	1.0	2.5	0.9	1.0	**37.2**	1.5
Piligrimage	0.1	0.2	0.3	0.4	**0.9**	0.2
Average	**40.0**	**51.1**	**40.5**	**45.4**	**48.8**	**44.8**

Gracz korzystający z najlepszej strategii

Game	MINI-1-S vs. MINI-Player
Connect4	52.41 [5.90]
Cephalopod Micro	45.92 [5.94]
Free for all 2P	48.33 [5.56]
Pentago	50.00 [5.96]
9 Board Tic-Tac-Toe	52.96 [5.75]
Connect4 Suicide	40.30 [5.60]
Checkers	57.78 [5.78]
Farming Quandries	58.89 [5.33]
Piligrimage	35.63 [5.33]
Average	**49.14** [5.68]

Podsumowanie

- Wprowadzenie strategii symulacji wyraźnie zwiększyło siłę gracza MiNI-Player
- Zmodyfikowana formuła wyboru ruchu EQ wraz ze strategiami również poprawia wyniki

Bez modyfikacji: wyniki vs CadiaPlayer to **1-8**

Z modyfikacjami: wyniki vs CadiaPlayer to **5-4**

Wciąż dużo otwartych pytań i problemów:

- **Modelowanie przeciwników**
- **Jak siła symulacji MCTS wpływa na jakość ostatecznego gracza?**
- **Uwzględnienie strategii przy wyborze ruchu do zagrania?**
  - W zależności od wiedzy o adekwatności strategii
  - W przypadku niedostatecznej liczby symulacji, żeby polegać na średnim wyniku

Wciąż dużo otwartych pytań i problemów:

- **Nowe strategie…**
- **Przyspieszenie symulacji – losowanie tylko podzbioru możliwych następnych stanów przy analizie ruchu przez strategie gry**
- **Podział gry na fazy early-mid-end i stosowanie odrębnych strategii**

Wciąż dużo otwartych pytań i problemów:

- **Monitorowanie wydajności strategii – ile razy „zadziałała” a ile razy każdy ruch miał taką samą ocenę (np. Mobility w Connect-4)**
- **Stosowanie strategii zastępczych do powyższych przypadków**
- **Pogrupowanie strategii w zestawy i połączenie z mechanizmem zrównoleglenia**

Dziękuję za uwagę