AI w grach: trenowanie prostych i wydajnych obliczeniowo modeli gry przy pomocy Offline Monte Carlo Tree Search

Maciej Świechowski maciej.swiechowski@qed.pl

Plan prezentacji

Krótko o Grailu

Monte Carlo Tree Search

Motywacja podejścia uczenia offline

Środowisko do eksperymentów

Algorytm uczenia – konwersji do drzewa decyzyjnego

Dużo wyników

Wnioski i podsumowanie

Pomysły na przyszłość


Grail

Grail

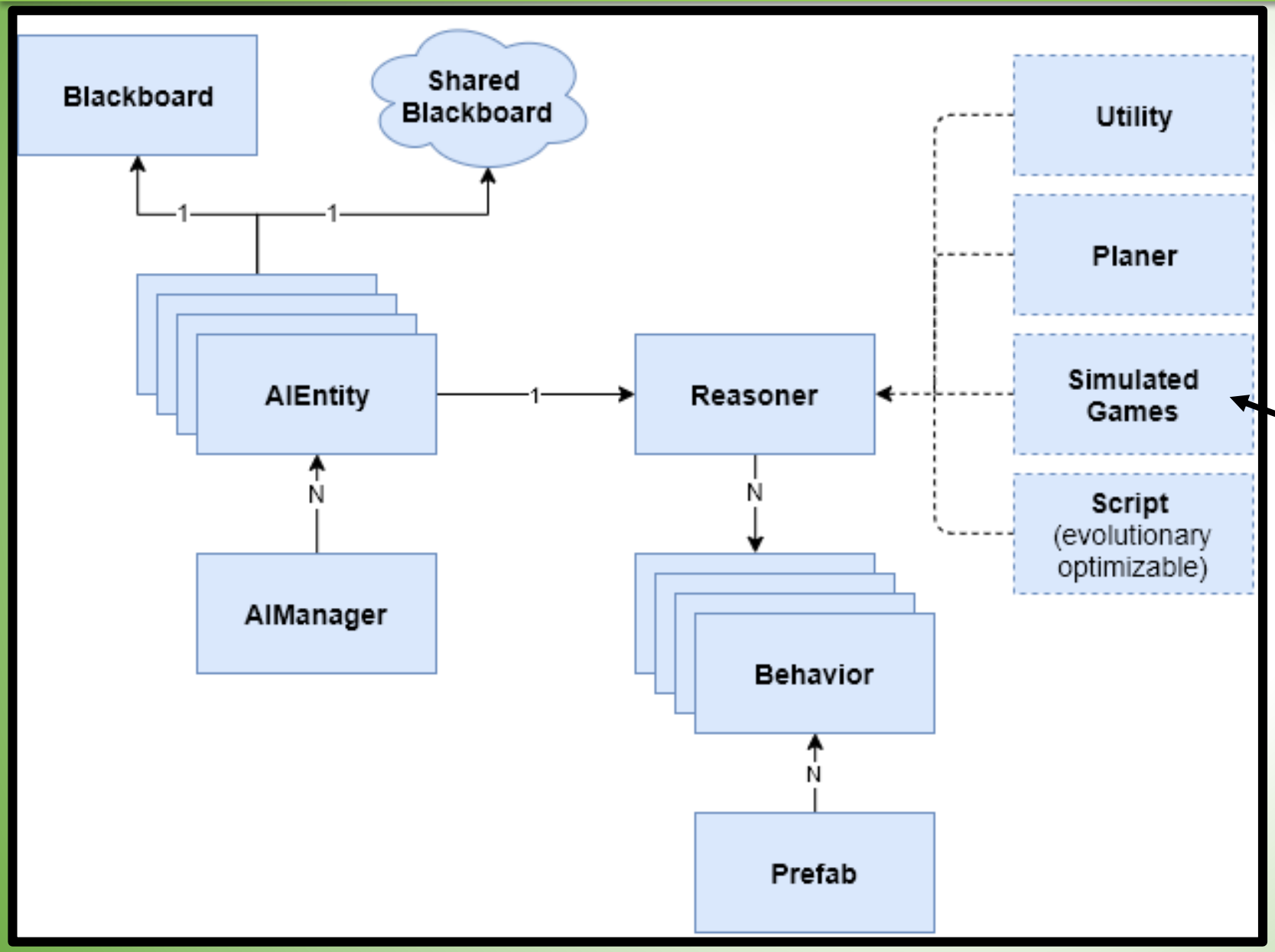
Oprogramowanie do tworzenia i konfiguracji AI dla twórców gier

- biblioteka w C++
- biblioteka w C#
- plugin do silnika Unity3D
- plugin do Unreal Engine
- zestaw narzędzi wspomagających opartych o GUI

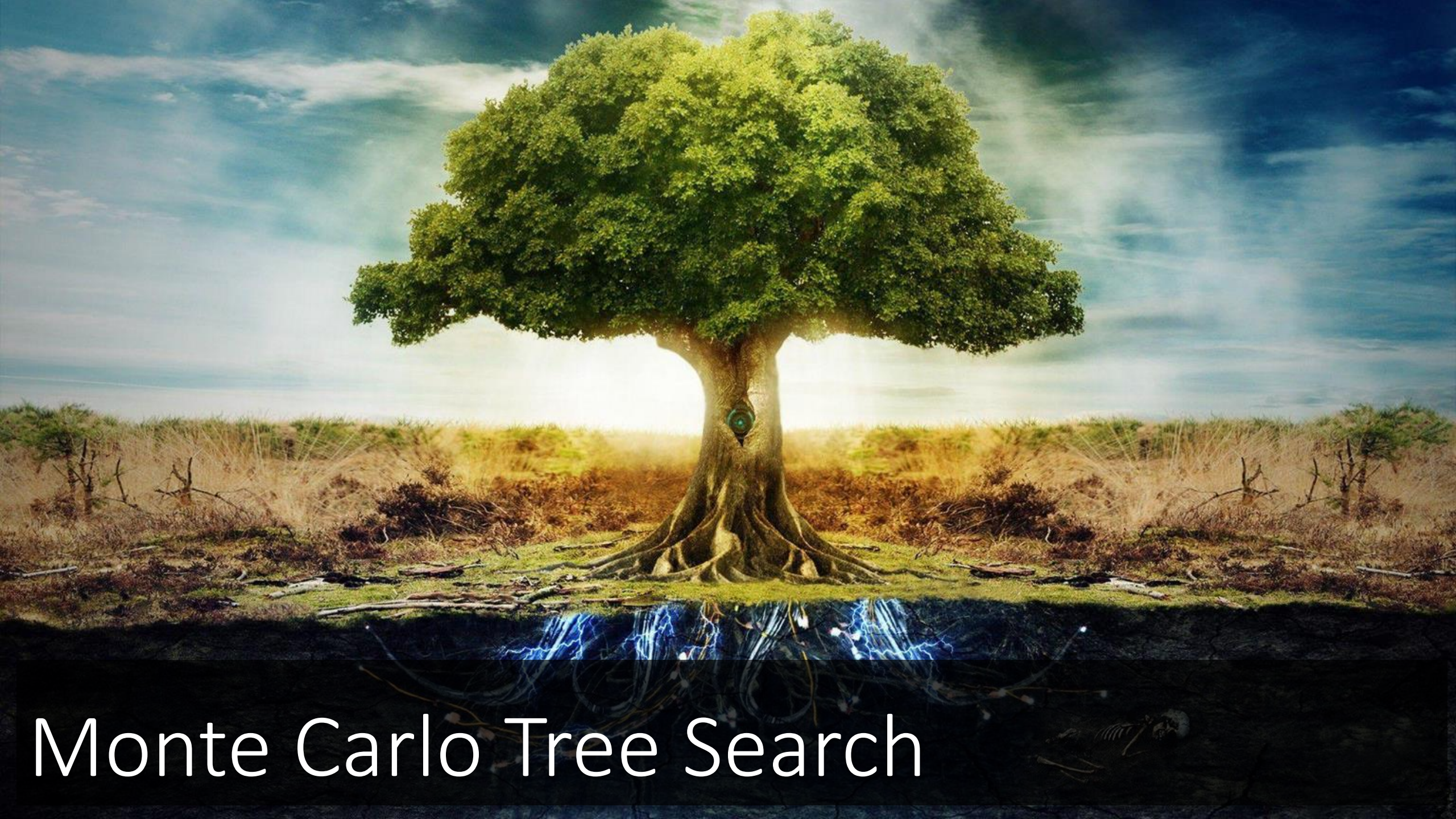
Wybrane aspekty silnika AI:

- system wieloagentowy
- komunikacja, reprezentacja i przepływ danych
- algorytmy wyboru zachowania
- diagnostyka, logowanie

Grail



dziś głównie o tym

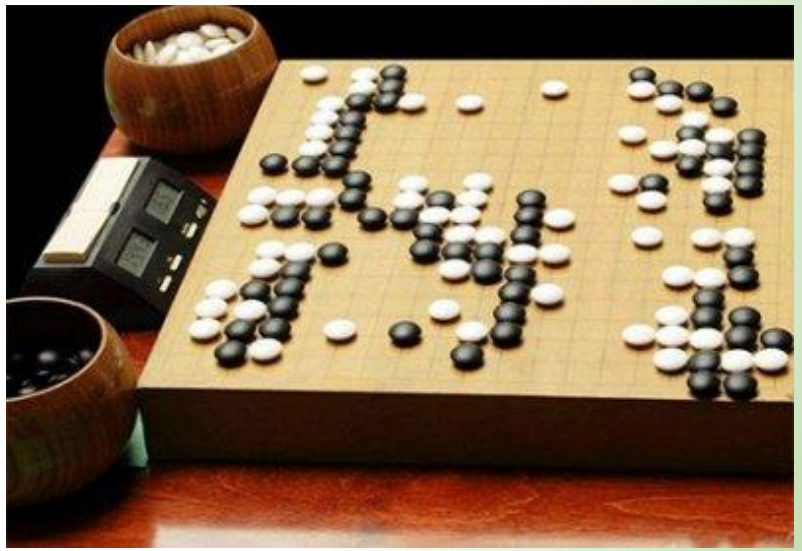


Monte Carlo Tree Search

Monte Carlo Tree-Search (MCTS)

Iteracyjne przeszukiwanie drzewa gry

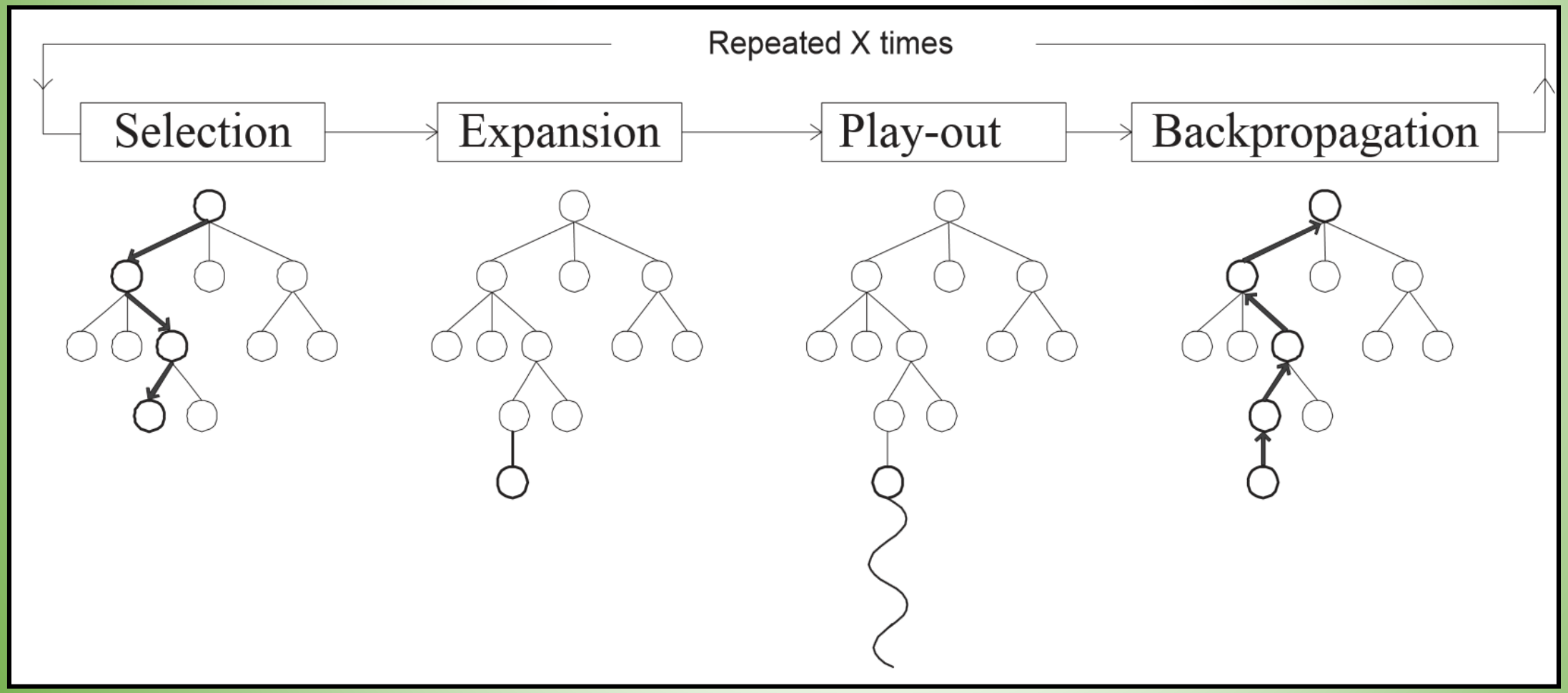
- wymagany jedynie (wydajny) symulator gry
- podstawa wielu przełomowych osiągnięć np. AlphaGo (Google DeepMind)



Monte Carlo Tree-Search (MCTS)

4 główne fazy (każda może mieć modyfikacje i rozszerzenia)

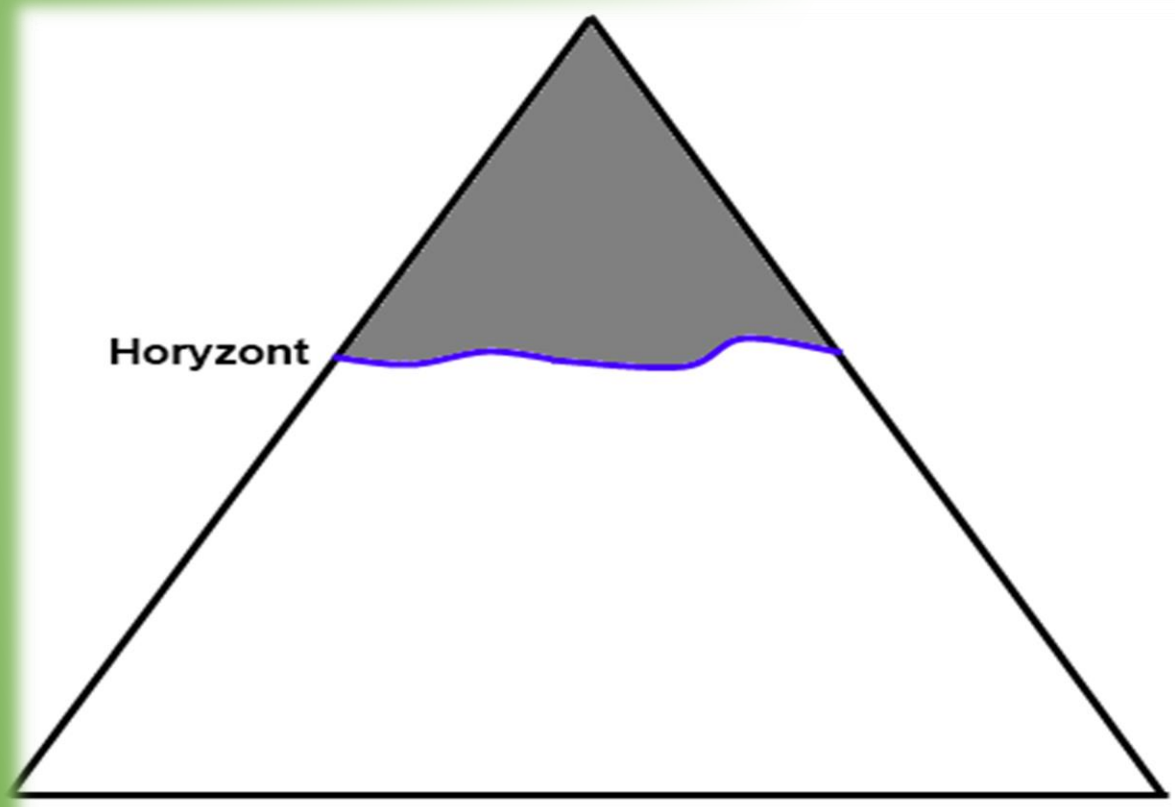
- metoda *aheurystyczna*



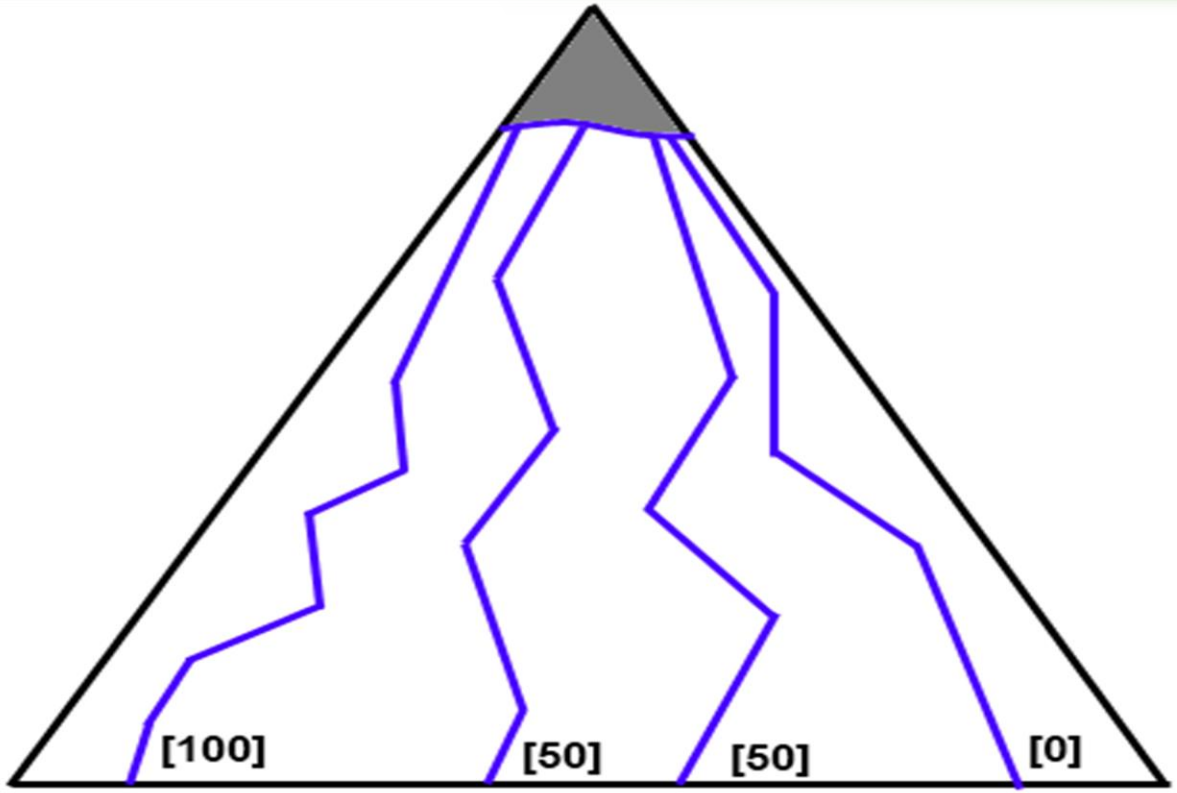
Monte Carlo Tree-Search (MCTS)

Metoda typu *anytime*

Bardziej niezbalansowane drzewo niż min-max / alfa-beta / MTD(f)



Przeszukiwanie drzewa gry
(ala min-max)



MCTS (ang. Monte Carlo Tree Search)

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) – średni 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***

Alternatywy dla UCT

ε-greedy

Comparison of Different Selection Strategies in Monte-Carlo Tree Search for the Game of Tron

Pierre Perick, David L. St-Pierre, Francis Maes and Damien Ernst

- Note that a Bernoulli random variable with $p = 0.5$ is the reward distribution that will give the highest variance (which is $\frac{1}{4}$).
- We can also compute the sample variance σ_j for each action.
- Then use the upper confidence bound for action j of:

$$\sqrt{\frac{\ln n}{n_j} \min\left(\frac{1}{4}, \left(\sigma_j + \frac{2 \ln n}{n_j}\right)\right)}$$

Modifications of UCT and sequence-like simulations for Monte-Carlo Go

Yizao Wang
Center of Applied Mathematics
Ecole Polytechnique, Palaiseau, France
yizao.wang@polytechnique.edu

Sylvain Gelly
TAO (INRIA), LRI, UMR (CNRS - Univ. Paris-Sud)
University of Paris-Sud, Orsay, France
sylvain.gelly@lri.fr

Wybór akcji w grze

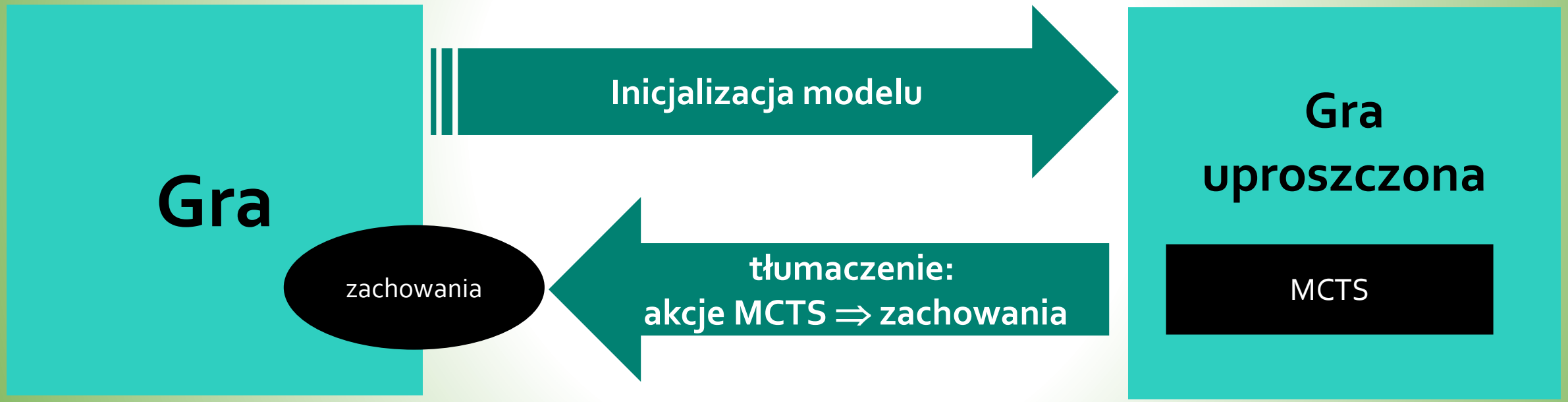
Metoda sterowana budżetem obliczeniowym

- liczba symulacji lub czas dostępny na obliczenia

Po wyczerpaniu budżetu, wybór akcji w grze:

$$a^* = \arg\max_{a \in A(s)} \{ Q(s,a) + \}$$

Gry uproszczone



Motywacja uczenia offline

Podejścia **symulacyjne są bardzo wymagające obliczeniowo**

W grach komputerowych, często budżet na AI jest niewielki

- od 2-3ms do 16ms na klatkę animacji
- nawet strategiczne decyzje – max kilka sekund
- gry czasu rzeczywistego – AI musi działać, czas goni, łatwa eksploatacja głupiego AI
- gry turowe – AI nie może wykonywać tury zbyt długo, by nie zniecierpliwić graczy

Podejścia symulacyjne bywają trudne w zrozumieniu i testowaniu

Podejścia symulacyjne bywają nieprzewidywalne (co ma wady i zalety)

Co jest standardem w branży?

Kiedyś:

- Skrypty, proste heurystyki
- Automaty skończone

Dzisiaj:

- Drzewa behawioralne

Jutro:

- ???

Drzewa behawioralne

De-facto grafowy i wizualny sposób definiowania skryptu

- Selektory i Sekwencje
- Dekoratory
- Węzły zachowań (behavior)

Drzewa behawioralne mogą modelować drzewa decyzyjne

- ale nie na odwrót

Wady drzew behawioralnych

Tworzone ręcznie => czasochłonny proces

Wymagają twórczej pracy projektanta AI

- same z siebie „nie rozwiązują AI”, nie optymalizują żadnej funkcji celu

Złożone AI może wymagać bardzo dużych i złożonych drzew

- znów, nie byłby to problem, gdyby były tworzone automatycznie

Praca z drzewami behawioralnymi źle się skaluje wraz ze zmianami w grze lub zwiększaniem złożoności gry

Motywacja uczenia offline

Motywacja, aby użyć MCTS jako trenera drzewa decyzyjnego

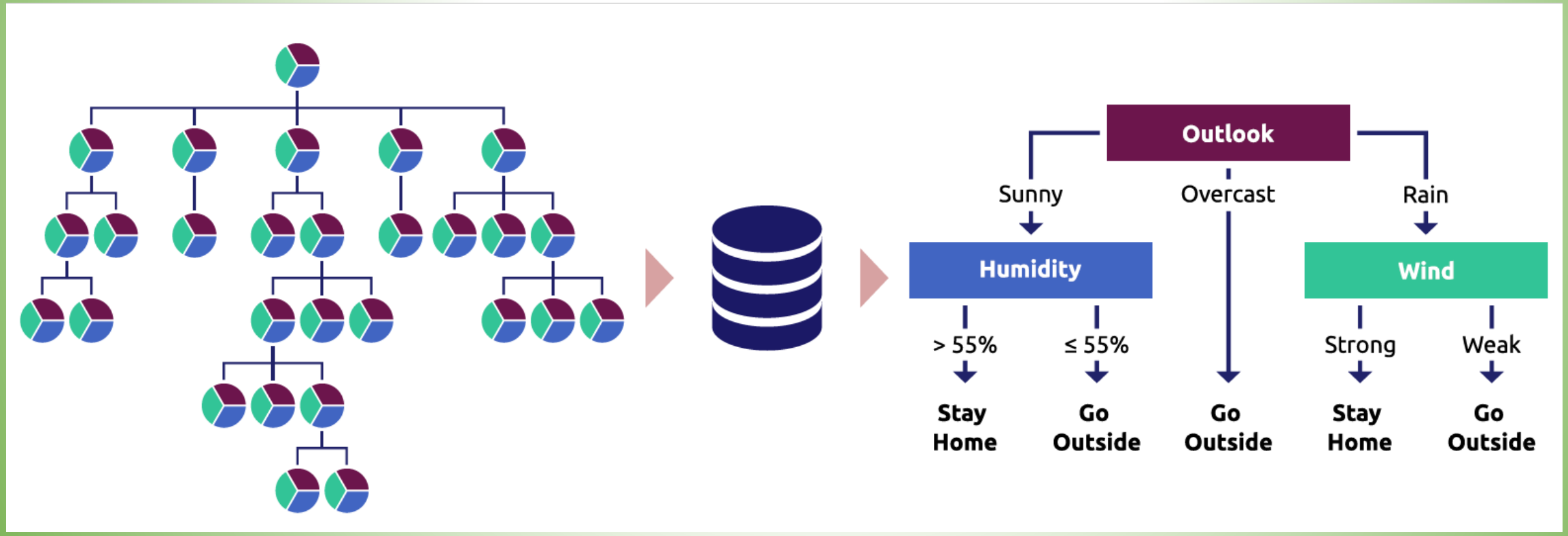
- minimalny narzut obliczeniowy
- łatwe testowanie
- przewidywalne i deterministyczne
- kojarzą się twórcom gier z drzewami behawioralnymi
- można ręcznie poprawić
- gdy potrzeba – można trenować tak, by „wygładzać zachowanie”.

  (MCTS działa bardziej wyrachowanie i asymptotycznie dąży do optymalnej gry)

Uczenie offline - metoda

Idea – kompresja wiedzy

- uruchamianie MCTS tylko w fazie produkcji gry
- tylko gry z botami
- **rozluźnione ograniczenia czasowe** np. 50 razy więcej czasu na decyzję

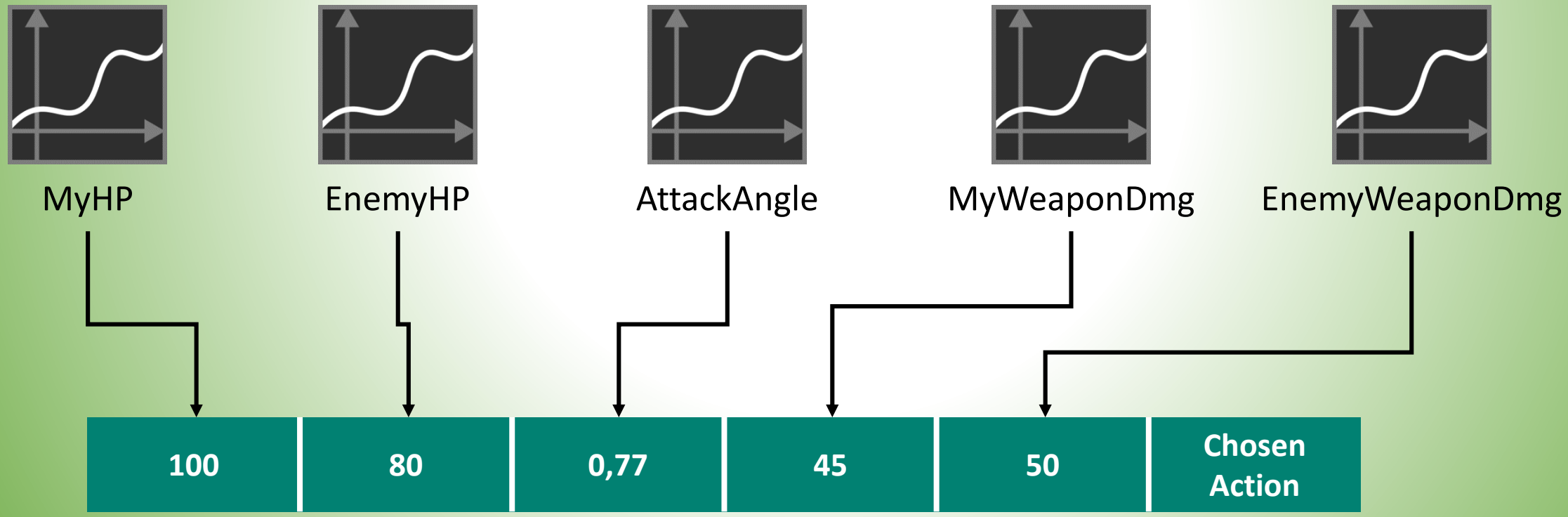


Wektoryzacja

Sprowadzenie stanu gry do wektora liczb zmiennoprzecinkowych

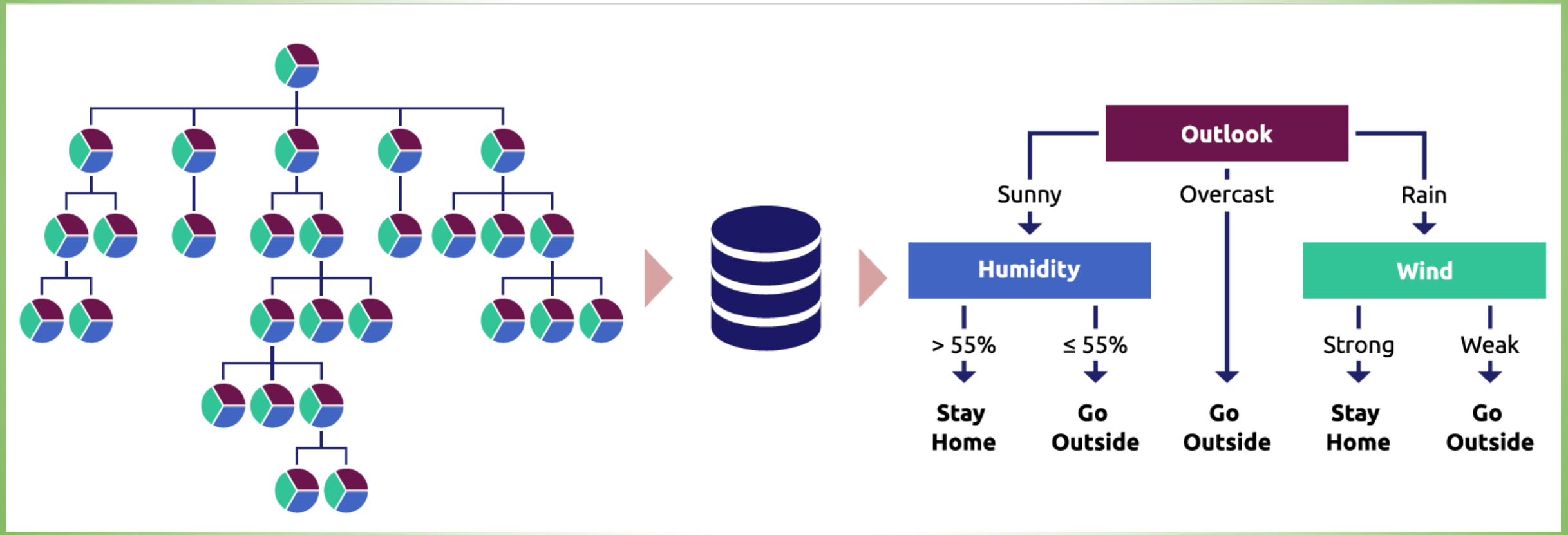
- w MCTS stan gry może być reprezentowany w dowolny sposób

Podobne zagadnienie jest w uczeniu maszynowym



Wektoryzacja

- elementy stanu jako czynniki, które wpływają na decyzje gracza



Kroki do wykonania

1. Przygotowanie wektoryzatora

2. Pętla zbierania danych (rozgrywki treningowe)
    a) Wektoryzacja węzłów MCTS
    b) Filtrowanie węzłów
    c) Agregacja po wektoryzacji
    d) Agregacja po akcji

3. Wygenerowanie drzewa decyzyjnego

Krok pętli zbierania danych

a) wektoryzacja węzłów MCTS

- gdy mamy stan gry; zapisanie węzła wraz z wektoryzacją

b) filtrowanie węzłów

- $node.visits \geq learningThreshold * totalVisits$

c) agregacja po wektoryzacji

- key: $[v_1,\ldots,v_n]$ value = list<node>
- operator porównania po współrzędnych z precyzją p = 0.01
- efektywne hashowanie

Krok pętli zbierania danych

b) filtrowanie węzłów

- $node.visits \geq learningThreshold * totalVisits$

Krok pętli zbierania danych

c) agregacja po wektoryzacji

- key: $[v_1,\ldots,v_n]$ value = list<node>
- operator porównania po współrzędnych z precyzją p = 0.01
- efektywne hashowanie

d) agregacja po akcji

Na wejściu mamy

- key: $[v_1,\ldots,v_n]$ value = list<node>

Dla każdego węzła na liście

- określamy najlepszą akcję $\boldsymbol{a}^*$ w węźle (argmax(Q))
- zapisujemy takie akcje w słowniku

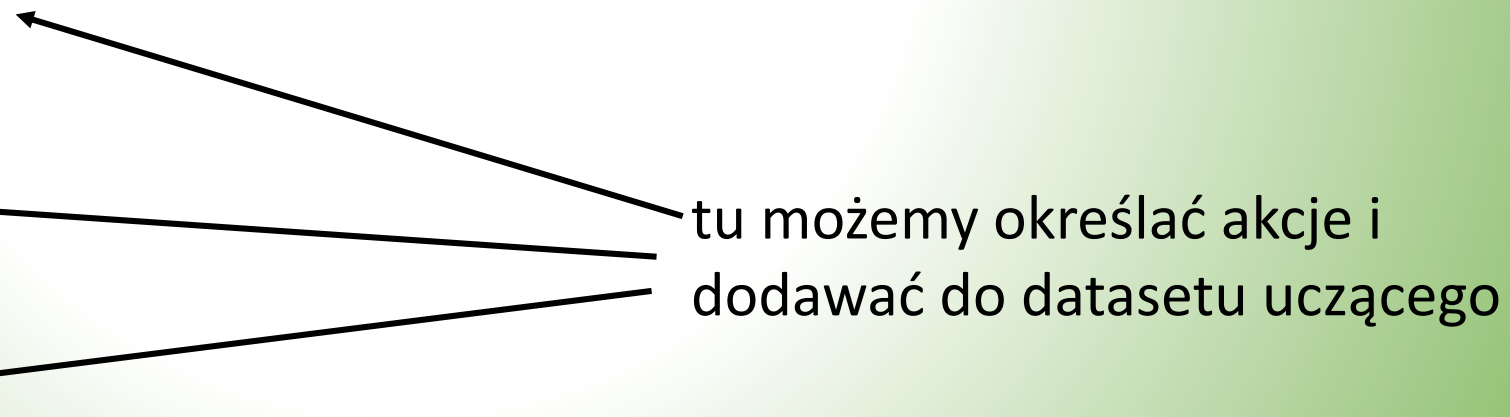
  key: [action] value = *action.visits*

Dla każdej najlepszej akcji $\boldsymbol{a}^*$ sumujemy liczbę odwiedzin

Akcja z największą liczbą odwiedzin wraz z wektoryzacją stanu dodawana jest do globalnego *datasetu* do uczenia drzewa decyzyjnego

Pętla zbierania danych

```
for k  trainingRepeats:
    for l gameRepeats:
        game.start()
        while(game not finished)
            mcts.run(iterations)
            playAction(mcts.getBestAction())
            [Pobranie danych?]
        end (while)
        [Pobranie danych?]
    end (for)
[Pobranie danych?]
end(for)
```



Drzewo decyzyjne

Uczone na podstawie zebranych danych w postaci

[wektoryzacja, wybrana akcja]

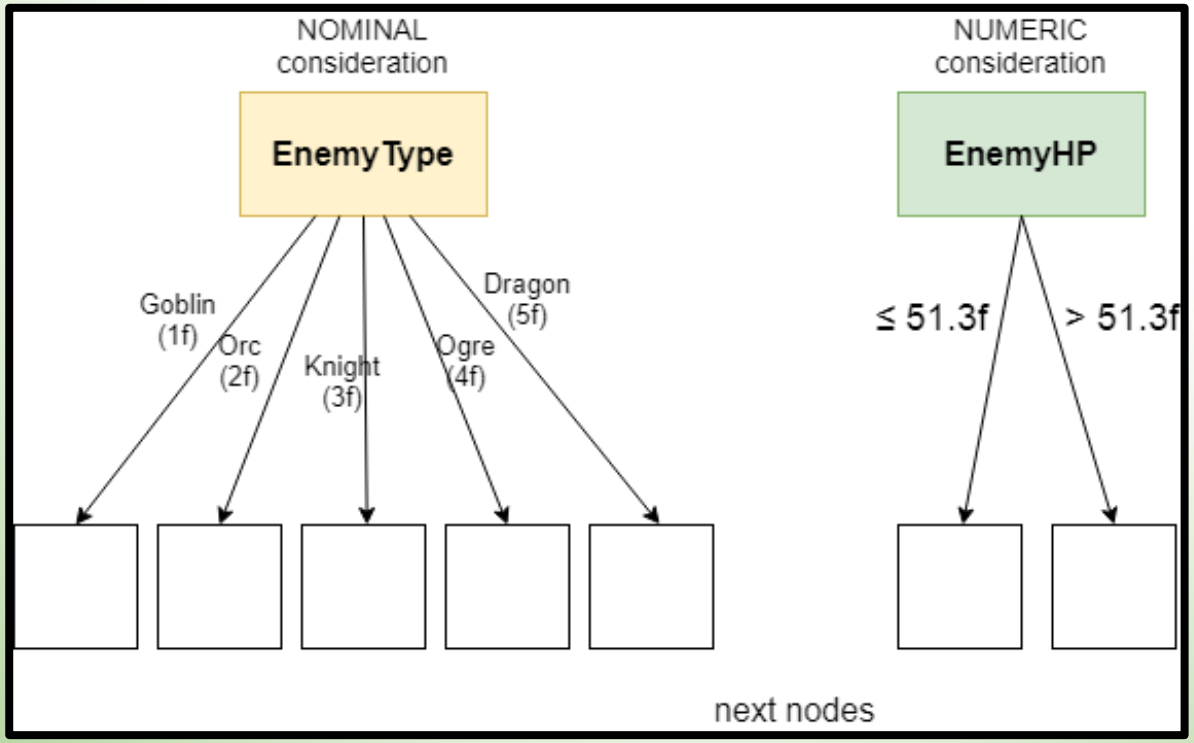
Do wygenerowania - algorytm **C4.5** (Ross Quinlan)

- ulepszenie algorytmu ID3 (Iterative Dichotomiser)
- bazuje na koncepcjach entropii oraz zysku informacji (*information gain*)

Drzewo decyzyjne

Ograniczenie do trzech rodzajów węzłów:

- nominalne: *IF(data[i] == value)*
- numeryczne, ciągłe: *IF(data[i] <= splitValue)* oraz *IF(data[i] > splitValue)*
- liście: zawierają decyzje (akcje gry)



Podsumowanie hiperparametrów

Parametry metody uczenia:

- liczba powtórzeń pętli
- sposób wektoryzacji
- learningThreshold – aby uczyć się na danych lepszej jakości
- moment agregacji akcji (wyboru najlepszej) i stworzenia próbki uczącej
  - STATE, GAME, N-GAMES

Parametry MCTS:

- współczynnik eksploracji (UCT constant)
- liczba iteracji

Środowisko do eksperymentów

Gra

Imitacja walki 1 vs. 1 w grach typu *rogue-like*

Gra uproszczona

W założeniu gra czasu rzeczywistego

Celowo coś nietypowego dla klasycznego MCTS

Główny protagonista, zwany dalej **graczem**

Przeciwnik: automatyczny robot, zwany dalej **przeciwnikiem**

Gra

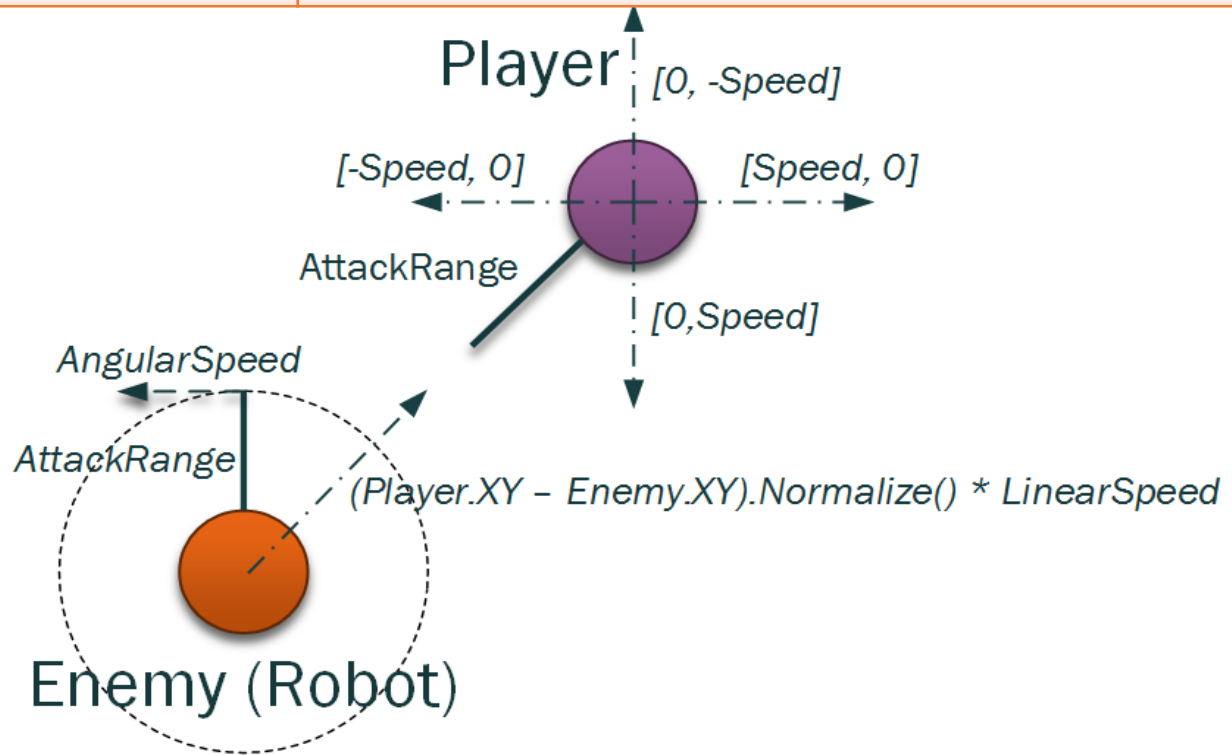
Gracz

- ruch w 4 kierunkach (WSAD)
- ruch zabiera czas
- atak zawsze w kierunku przeciwnika, legalny tylko gdy ten jest w zasięgu

Przeciwnik

- ruch zawsze w kierunku głównego gracza
- ruch w tym samym czasie, co ruch gracza
- obracające się ramię, które atakuje automatycznie poprzez kolizję z nim

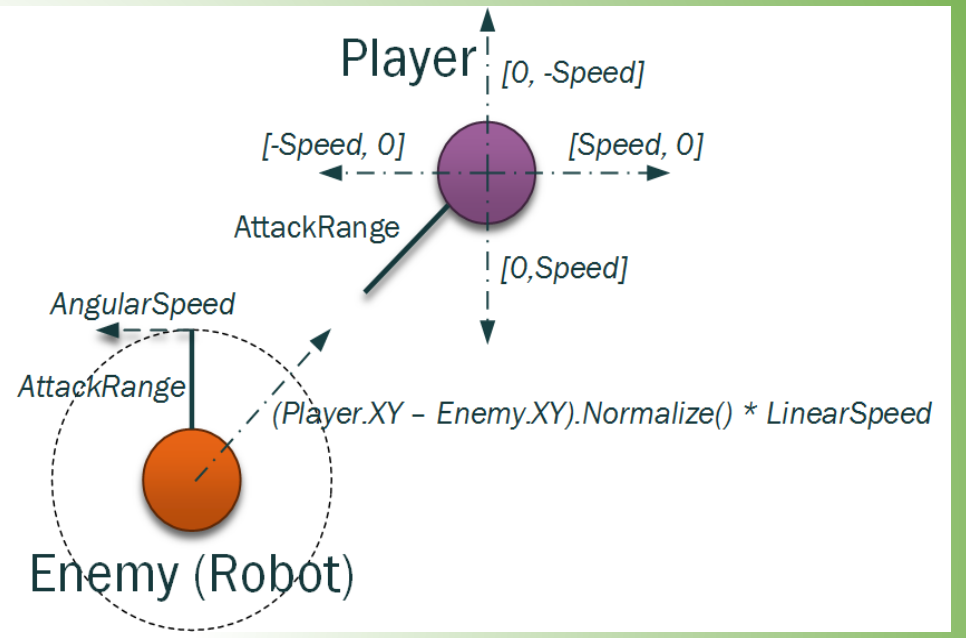
Parametr	Interpretacja	Wstępna wartość
PlayerSpeed	liczba jednostek odległości, którą porusza się gracz w ciągu 1 sekundy	2.2
PlayerAttackRange	zasięg ataku gracza w jednostkach odległości	20.0
EnemyLinearSpeed	liczba jednostek odległości, którą porusza się przeciwnik w ciągu 1 sekundy	1.2
EnemyAngularSpeed	kąt w radianach, o który obraca się ramie przeciwnika w ciągu 1 sekundy	$\Pi/4$
EnemyAttackRange	zasięg ataku przeciwnika w jednostkach odległości	20.0
BodyRadius	rozmiar graczy do określania kolizji	10.0
Target Points	liczba punktów do zdobycia przez gracza, aby wygrać (liczba potrzebnych trafień)	36
Max Time	liczba sekund gry, po której gracz przegrywa	150



Gra - atakowanie

Atak głównego protagonisty

- tylko, gdy w zasięgu (distance ≤ AttackRange)
- atak zawsze w kierunku przeciwnika
- nie „kosztuje” czasu…
- … ale nie można atakować 2 razy bezpośrednio po sobie
- skuteczny daje +1 punkt



Atak przeciwnika

- ma ramie, obraca się automatycznie
- kolizja Enemy Arm – Player Body => wygrana przeciwnika
- kolizja Enemy Body – Player Body => wygrana przeciwnika
- Brak wygranej gracza w ciągu 150 sekund => wygrana przeciwnika

Wyniki

Wstęp

Każdy eksperyment to uśredniony wynik **204 powtórzeń**, o ile nie będzie explicite powiedziane inaczej.

$$wynik\ gry = \begin{cases} 1, & jeżeli\ gracz\ wygrał \\ 0, & jeżeli\ gracz\ przegrał \end{cases}$$

1. Eksperymenty kalibracyjne
2. Eksperymenty związane z oceną offline learning

Zobaczmy jak trudna to gra

Attack Range	Player Speed							
	1,2	1,40	1,6	1,8	1,9	2	2,2	2,8
12	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,39
14	0,00	0,00	0,00	0,00	0,00	0,10	0,19	0,94
16	0,00	0,00	0,00	0,01	0,09	0,35	0,79	1,00
18	0,00	0,00	0,00	0,37	0,62	0,81	0,97	1,00
20	0,00	0,00	0,30	0,84	0,93	0,99	1,00	1,00
26	0,57	0,98	1,00	1,00	1,00	1,00	1,00	1,00

Parametr	Wartość
PlayerSpeed	$\rightarrow$ 2.0
PlayerAttackRange	$\rightarrow$ 18
EnemyLinearSpeed	1.2
EnemyAngularSpeed	$\prod/4$
EnemyAttackRange	20.0

Parametr	Wartość
BodyRadius	10.0
Target Points	36
Max Time	150
UCT constant	$\sqrt{2}$
MCTS iterations	50 000

Tuning parametru eksploracji - UCT constant

UCT-C	Score	UCT-C	Score	UCT-C	Score
0	0.76	$0.5 * \sqrt{(2)}$	0.84	$16 * \sqrt{2}$	0.83
$0.01 * \sqrt{(2)}$	0.79	$\sqrt{2}$	0.81	$32 * \sqrt{2}$	0.82
0.1	0.84	$2 * \sqrt{2}$	0.80	$64 * \sqrt{2}$	0.81
$0.1 * \sqrt{(2)}$	0.88	$4 * \sqrt{2}$	0.81	$128 * \sqrt{2}$	0.79
$0.25 * \sqrt{(2)}$	0.81	$8 * \sqrt{2}$	0.81		

Parametr	Wartość
PlayerSpeed	2.0
PlayerAttackRange	18
EnemyLinearSpeed	1.2
EnemyAngularSpeed	$\prod/4$
EnemyAttackRange	20.0

Parametr	Wartość
BodyRadius	10.0
Target Points	36
Max Time	150
UCT constant	$\rightarrow 0.1 * \sqrt{2}$
MCTS iterations	50 000

Tuning liczby iteracji

Iterations	Score	Iterations	Score	Iterations	Score
$1K$	0.00	$40K$	0.74	$150K$	0.99
$5K$	0.06	$50K$	0.88	$200K$	0.99
$10K$	0.28	$75K$	0.88	$250K$	1.00
$20K$	0.42	$100K$	0.95	$300K$	1.00
$30K$	0.59	$125K$	0.97	$400K$	1.00

dla 100K iteracji
baseline = 0.95

Parametr	Wartość
PlayerSpeed	2.0
PlayerAttackRange	18
EnemyLinearSpeed	1.2
EnemyAngularSpeed	$\prod/4$
EnemyAttackRange	20.0

Parametr	Wartość
BodyRadius	10.0
Target Points	36
Max Time	150
UCT constant	$0.1 * \sqrt{2}$
MCTS iterations	$\rightarrow$ 100 000

Główne eksperymenty

Moment na określenie sposobu wektoryzacji stanu

Wektoryzator powinien **uogólniać stan**

- nie chcemy np. absolutnej pozycji
- nie chcemy, aby było zbyt dużo unikalnych stanów
  - to będzie skutkować mało użytecznym drzewem decyzyjnym
- jednocześnie, wektoryzator musi brać ważne czynniki pod uwagę pod kątem wyboru akcji przez gracza

Wektoryzator złożony z 4 elementów

- ***Distance***
- ***EnemyAngle***
- ***RelativeAngle***
- ***GamePhase***

Elementy wektoryzatora (1,2)

- ***Distance***: odległość między graczem a przeciwnikiem
  - potrzebne do decydowania o ataku i ucieczce
  - liczona od środków kół reprezentujących ciała postaci

- ***EnemyAngle:*** bieżąca wartość kąta ramienia przeciwnika
  - ma duży wpływa na to, czy przeciwnik nas trafi
  - część gry, a więc parametr dostępny od ręki

Elementy wektoryzatora (3,4)

- ***RelativeAngle:*** względne ustawienie - kąt, jaki tworzy wektor gracz – przeciwnik z arbitralnie wybranym kierunkiem odniesienia

$$dir = \|PlayerPosition - EnemyPosition\|$$

$$RelativeAngle = \begin{cases} Atan2(dir.Y, dir.X), & Atan2(dir.Y, dir.X) \geq 0 \\ Atan2(dir.Y, dir.X) + 2\pi, & Atan2(dir.Y, dir.X) < 0 \end{cases}$$

- ***GamePhase:*** ”one-hot encoding” czy distance < 35, żeby odróżnić fazę długiego zbliżania się do przeciwnika od powtarzalnego procesu walki

Dyskretyzacja kątów

Wektoryzator będziemy parametryzować dwiema liczbami

DT(A, B) – drzewo decyzyjne z granulacją wektoryzacji A oraz B, co oznacza

$$\text{vectorization}[1] = \left\lfloor \frac{RelativeAngle * 180}{A * \pi} \right\rfloor$$

$$\text{vectorization}[2] = \left\lfloor \frac{EnemyAngle * 180}{B * \pi} \right\rfloor$$

We wstępnych testach, najlepiej radziło sobie **DT(40, 40)**

Uczenie drzewa – sposób agregacji akcji

1. `STATE` – po każdym stanie
2. `GAME` – po każdej grze
3. `N-GAMES` - po wykonaniu n gier

```
for k  trainingRepeats:
    for l gameRepeats:
        game.start()
        while(game not finished)
                mcts.run(iterations)
                playAction(mcts.getBestAction())
                [Pobranie danych?]
        end (while)
            [Pobranie danych?]
    end (for)
[Pobranie danych?]
end(for)
```

Uczenie drzewa – sposób agregacji akcji

Training repeats	Data aggregation method	Learning threshold 0	0.01	0.1	0.5	0.99
1	$STATE$	0.00	0.00	0.5	0.52	0.54
1	$GAME$	0.00	0.00	0.5	0.63	0.63
1	$10-GAMES$	0.00	0.00	0.48	0.62	0.64
10	$STATE$	0.00	0.00	0.46	0.51	0.52
10	$GAME$	0.00	0.00	0.54	0.63	0.64
10	$10-GAMES$	0.00	0.00	0.53	0.63	0.64
50	$STATE$					0.50
50	$GAME$					0.60
50	$10-GAMES$					0.65

Wnioski:

- spodziewałem się dużo lepszych wyników
- dużo gorzej niż 0.95
- GAME najlepszą opcją
- zwiększanie powtórzeń nie zwiększa oczekiwanego wyniku *(później okaże się o co chodzi!)*
- learning threshold kluczowy
- nie opłaca uczyć się na całym drzewie MCTS, tylko blisko korzenia

Wynik gry + accuracy na zbiorze treningowym

Wyniki

Training repeats	Data conversion method	Learning threshold 0	0.01	0.1	0.5	0.99
1	$STATE$	0.00	0.00	0.5	0.52	0.54
1	$GAME$	0.00	0.00	0.5	0.63	0.63
1	$10 - GAMES$	0.00	0.00	0.48	0.62	0.64

Accuracy

Training repeats	Data conversion method	Learning threshold 0	0.01	0.1	0.5	0.99
1	$STATE$	0.90	0.91	0.541	0.61	0.61
1	$GAME$	0.90	0.89	0.54	0.61	0.62
1	$10 - GAMES$	0.91	0.90	0.59	0.62	0.63

Ustalenie hiperparametrów

Training repeats	Data conversion method	Learning threshold 0	0.01	0.1	0.5	0.99
1	$STATE$	0.00	0.00	0.5	0.52	0.54
1	$GAME$	0.00	0.00	0.5	0.63	0.63
1	$10-GAMES$	0.00	0.00	0.48	0.62	0.64

Wybrano metodę nauki *GAME:*

- określanie najlepszych akcji w węzłach po 1 pełnej grze
- metoda porównywalnie dobra z 10-GAMES, a 10 razy szybsza
- learningThreshold = 0.99

Test 1-1 (score)

Jeden, ustalony stan startowy podczas trenowania i testów

Relative Angle	Enemy Angle										
	1	5	7	10	20	30	40	60	90	120	180
1	0,95	0,95	0,87	0,57	0,21	0,17	0,25	0,17	0,15	0,11	0,08
5	0,95	0,93	0,76	0,51			0,40			0,27	0,19
10	0,96	0,96	0,76	0,50			0,48			0,26	0,26
20	0,95	0,94	0,77	0,59	0,48	0,56	0,57	0,46			0,28
30	0,97	0,93	0,72	0,57	0,54	0,6	0,50		0,46		0,28
40	0,98	0,93	0,71	0,54	0,60	0,64	0,64	0,55		0,36	0,30
60	0,97	0,93	0,75	0,55	0,54	0,66	0,60	0,57	0,54		0,30
90	0,96	0,94	0,76	0,58	0,58	0,66	0,58	0,62	0,48		0,32
120	0,96	0,94	0,74	0,56	0,57	0,65	0,67	0,52		0,40	0,32
180	0,95	0,94	0,74	0,55	0,63	0,64	0,65	0,59	0,50	0,42	0,30

wektoryzator (40, 40) okazał się nie być globalnym maksimum

Test 1-1 (acc)

Accuracy na zbiorze treningowym

Relative Angle	Enemy Angle										
	1	5	7	10	20	30	40	60	90	120	180
1	1,00	1,00	0,99	0,98	0,95	0,94	0,91	0,86	0,81	0,75	0,71
5	1,00	0,99	0,98	0,92			0,77			0,60	0,56
10	1,00	1,00	0,98	0,89			0,69			0,55	0,52
20	1,00	1,00	0,97	0,85	0,73	0,67	0,62	0,57			0,48
30	1,00	1,00	0,97	0,84	0,72	0,63	0,61		0,50		0,46
40	1,00	0,99	0,97	0,84	0,70	0,65	0,63	0,56		0,51	0,46
60	1,00	0,99	0,97	0,84	0,68	0,62	0,59	0,54	0,49		0,45
90	1,00	0,99	0,97	0,83	0,67	0,61	0,58	0,53	0,49		0,45
120	1,00	0,99	0,97	0,81	0,67	0,61	0,59	0,53		0,47	0,46
180	1,00	0,99	0,97	0,82	0,60	0,60	0,59	0,53	0,49	0,47	0,46

bardzo przejrzysta zależność

Test 1-R

Trening na tym samym 1 stanie startowym co ostatnio

Test na losowych stanach próbkowanych z okręgu punktów:

- środkiem okręgu gracz
- promień okręgu = 70
- poprzedni ustalony stan startowy też ma odległość = 70, więc leży na tym okręgu
- nie trzeba zmieniać max czasu gry

Tym razem aż 2004 powtórzeń eksperymentu (gier).

- ze względu na większą losowość

Test 1-R

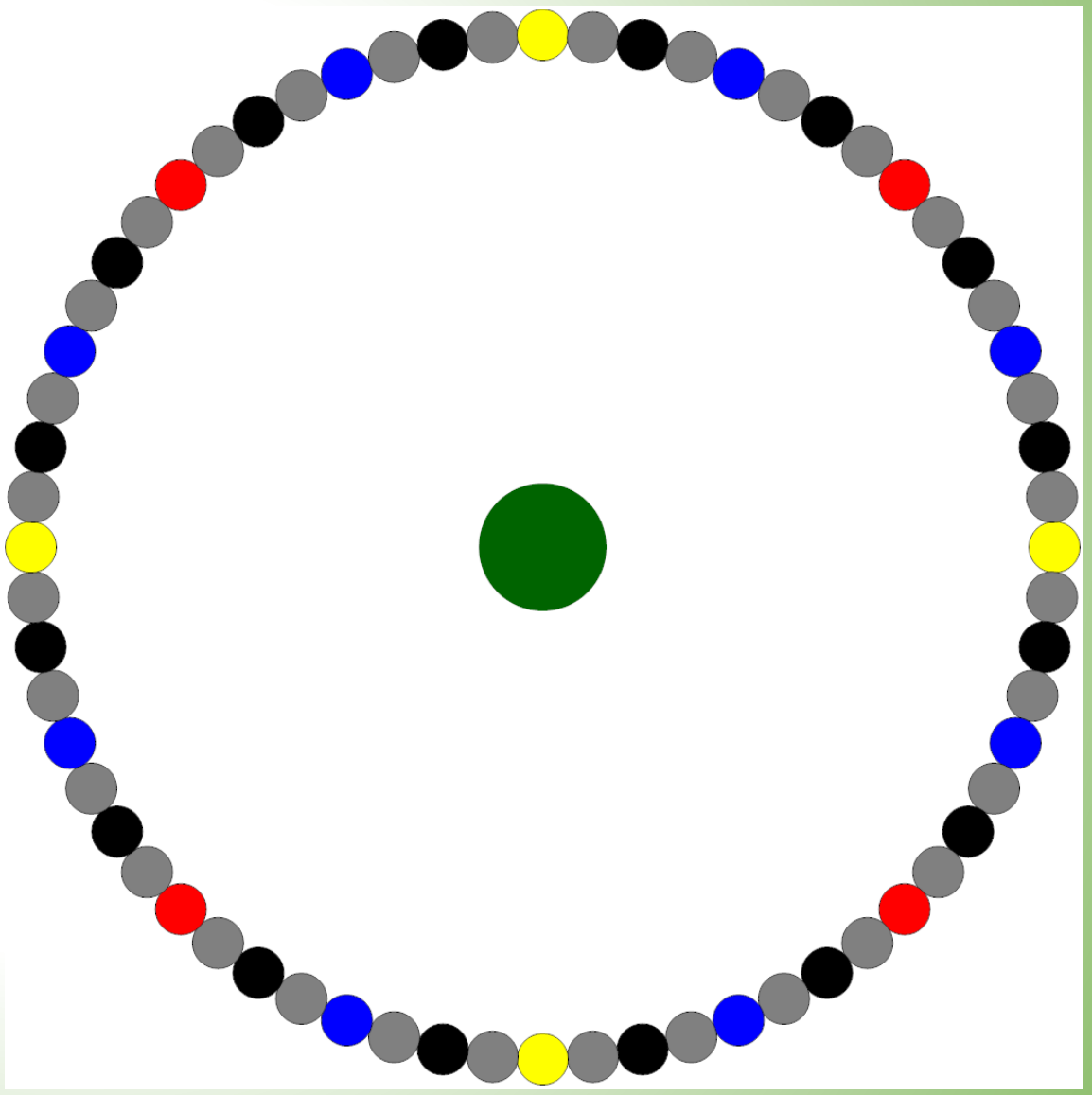
Relative Angle	Enemy Angle										
	1	5	7	10	20	30	40	60	90	120	180
1	0,01	0,02	0,02	0,04			0,09		0,09	0,08	0,08
5		0,01									
10		0,01		0,04			0,17				
20			0,02		0,12				0,19		0,15
30							0,23	0,23			
40	0,01	0,02	0,02			0,22	0,23	0,26	0,24	0,19	0,19
60						0,20	0,24	0,21	0,23	0,21	
90	0,01	0,01	0,01			0,24	0,24	0,24	0,27	0,20	
120					0,14	0,24	0,25	0,23	0,24	0,21	
180	0,01		0,01	0,05	0,15	0,23	0,26	0,23	0,24	0,25	0,21

wniosek: był duży overfitting do ustalonego stanu początkowego

accuracy na zbiorze treningowym się nie zmieniło

Lepsze pozycje startowe

Liczba pozycji	Które kolory
4	żółte
8	żółte + czerwone
16	żółte + czerwone + niebieskie
32	szare
64	wszystkie



Wynik gry

Relative Angle	Enemy Angle	Score T-16-R	Accuracy T-16-R	ΔScore T-1-1	ΔScore T-1-R
90	30	0.73	0.49	0.07	0.49
120	40	0.69	0.71	0.02	0.44
90	90	0.66	0.43	0.18	0.39
40	40	0.65	0.69	0.00	0.42
40	60	0.65	0.55	0.10	0.39
90	40	0.65	0.51	0.07	0.41
1	1	0.64	0.99	−0.31	0.63
180	40	0.63	0.41	−0.02	0.37
60	60	0.60	0.48	0.03	0.36
5	5	0.57	0.98	−0.36	0.56
20	20	0.36	0.71	−0.12	0.24
40	1	0.33	0.78	−0.65	0.32
180	120	0.28	0.39	−0.14	0.03
40	10	0.26	0.70	−0.28	0.22
10	10	0.25	0.88	−0.25	0.21
90	7	0.25	0.62	−0.51	0.24
20	180	0.19	0.48	−0.09	0.04

- Trening na 16 stanach
- Test na losowych stanach
- Próbka: 2004 gier (powtórzeń)
- Wybrane wektoryzatory

Pokazana zmiana w porównaniu do:

- T-1-1: test i trening na jednym, ustalonym stanie
- T-1-R: trening na jednym, ustalonym stanie i test na losowych stanach

Liczne wnioski

Wynik gry

Relative Angle	Enemy Angle	Unique training states 4	8	16	32	64
1	1	0.10	0.52	0.90	0.91	0.92
90	30	0.52	0.70	0.76	0.85	0.85
90	90	0.58	0.66	0.74	0.83	0.90
180	40	0.41	0.48	0.64	0.77	0.89

- Trening na podzbiorze koła stanów
- Test na 16 stanach
- Wybrane „ciekawe” wektoryzatory z różnych kategorii

Wnioski:

- uczenie na większym zbiorze stanów niż startowe ma sens!
- uczenie się na różnorodnych stanach kluczem!
- drzewo decyzyjne musi inaczej eksplorować przestrzeń niż MCTS, który je nauczył
- stopień poprawy zależy od wektoryzatora
- niektóre wektoryzatory potrzebują więcej stanów
- …

Generalizacja

Target Points	MCTS-100K	DT(90,30)	DT(1,1)
20	**1.00**	0.85	0.92
24	**1.00**	0.85	0.92
28	**1.00**	0.85	0.91
32	**0.99**	0.85	0.91
36	**0.95**	0.85	0.91
38	0.83	0.85	**0.88**
40	0.68	0.84	**0.87**
42	0.39	0.84	**0.86**
46	0.01	0.83	**0.84**
50	0.00	**0.83**	0.82
80	0.00	**0.83**	0.79

Wnioski:

- MCTS jest skuteczniejszy o ile budżet obliczeniowy pozwala na znalezienie rozwiązania
- MCTS się kiepsko skaluje w problemach przestrzennych i środowiskach takich jak gra testowa
- w odpowiednio ogólnym problemie (np. powtarzalnym zachowaniu) drzewo decyzyjne ma własność generalizacji
- drzewo decyzyjne, gdy już działa, to świetnie się skaluje

Wydajność

Czas na podjęcie pojedynczej decyzji w grze

MCTS-Iterations	MCTS	DT(90,30)	DT(1,1)
$25K$	220ms	$< 1ms$	$< 1ms$
$50K$	420ms	$< 1ms$	$< 1ms$
$100K$	792ms	$< 1ms$	$< 1ms$
$200K$	1598ms	$< 1ms$	$< 1ms$

Podsumowanie

Wnioski

- Metoda ma duży potencjał, lecz wymaga sporo przygotowań („setupu”) oraz pewnej świadomości jej działania
- Ryzyko przeuczenia, overfitting
- Wektoryzacja ma ogromny wpływ
- Nie dla każdej gry
- Uczenie na wielu unikalnych / różnorodnych sytuacjach
- Drzewo decyzyjne dobrze się uogólnia
  - może nawet uzyskiwać lepszy wynik niż MCTS, który je nauczył
- Wyprodukowane drzewo jest bardzo szybkie w działaniu
- Ciekawe wyniki

Możliwy rozwój

Zastosowanie technik **baggingu** i **boostingu**

- w szczególności **lasów losowych**

Uczenie przy pomocy MCTS zupełnie **czegoś innego niż drzewo decyzyjne**

- uwaga – można stracić klarowność, testowalność i wyjaśnialność

Sprawdzenie w **innych środowiskach testowych**

Szacowanie błędu, który powstaje przy konwersji między postaciami

Próba **automatycznego tworzenia wektoryzatorów**

Kontakt

www.grail.com.pl
www.grail.com.pl/documentation

Piotr Biczyk
piotr.biczyk@qed.pl

Maciej Świechowski
maciej.swiechowski@qed.pl

QED SOFTWARE