Monte Carlo Tree Search

Marika Ivanová

23. 10. 2012

Obsah

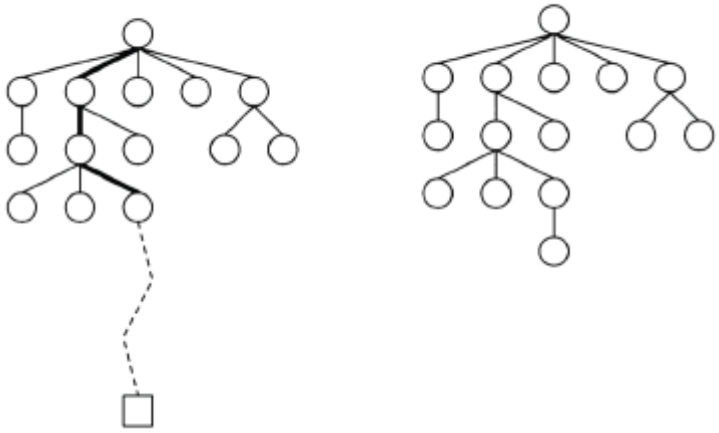
- Představení
  - Základní vlastnosti
  - Oblast použití
  - Bandit Problem
- Popis metody
  - Algoritmus
  - UCT
  - Charakteristika
  - Terminologie
- Vylepšení a Heuristiky
- Aplikace

Vlastnosti

- Metoda hledání optimálních rozhodnutí
- Zkoumáním náhodných vzorků postupně buduje prohledávací strom
- Rozšířeno u her s náhodnými prvky nebo částečně pozorovatelným světem
- Překvapivý úspěch u některých deterministických her s úplnou informací
  - 1993: První aplikace MC metod ve hře Go [1]
  - 2006: Mimořádně úspěšná aplikace MCTS v Go[2]
  - 2009: MoHex vítěz turnaje ve hře Hex [3]

Vlastnosti

- Best-first search
- Nepotřebuje doménově závislou heuristiku
- Základní algoritmus iterativně buduje prohledávací strom, dokud není dosaženo stanoveného limitu
- Dle zvolené strategie se určí nejvhodnější vrchol aktuálního stromu
- Simulace probíhá z vybraného vrcholu až do dosažení koncového stavu
- Aktualizace stromu na základě výsledku simulace



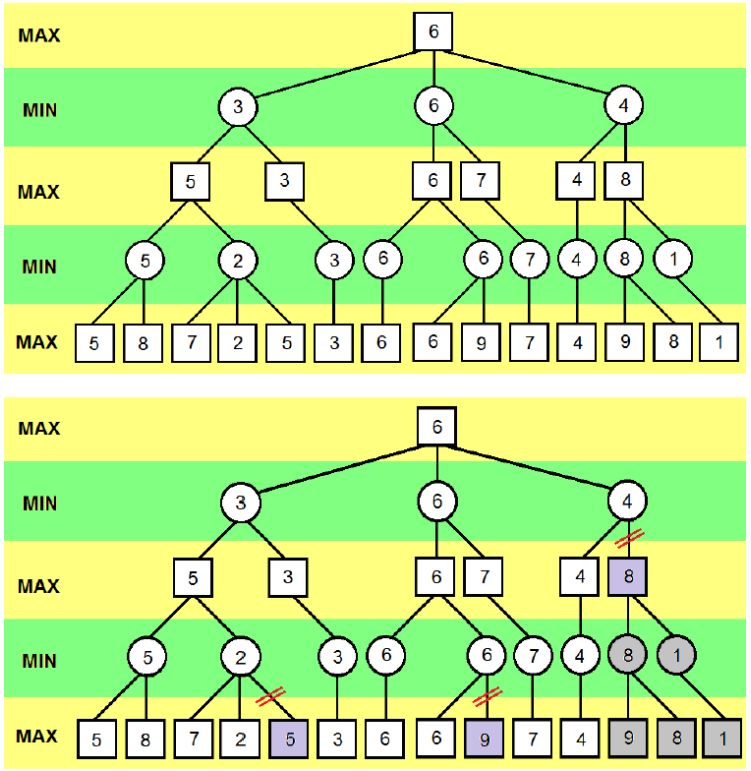
Důvod použití

- Máme přeci α-β s mnoha heuristikami
  - Iterative deepening
  - Killer heuristic
  - Null move heuristic
  - History heuristic
  - Transposition tables
  - …

- Šachové počítače porážejí velmistry
  - Kasparov vs. Deep Blue (1996)

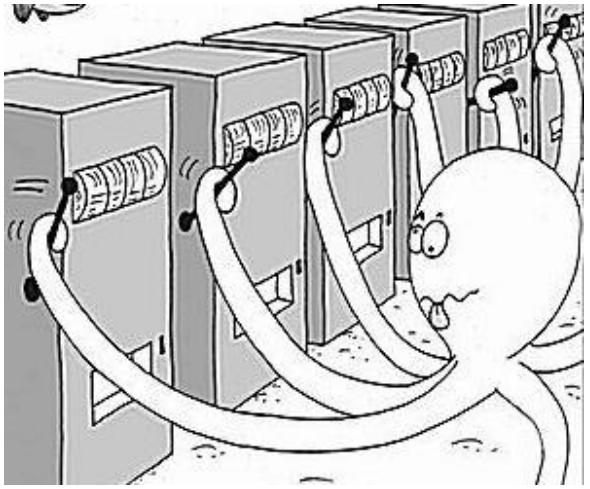
Důvod použití

- U některých her jsou standardní postupy neúspěšné
  - Příliš velký stavový prostor
    - Šachy: 10^{126} Go: 10^{360}
  - Obtížné ohodnocování
    - Závisí na konkrétní aplikaci
  - Náročná pravidla prořezávání



Bandit problem

- K-armed bandit je problém strojového učení
- *K* automatů na mince, vždy jeden vybereme a dostaneme odměnu dle distribuce přiřazené každému automatu
- Cílem je maximalizovat celkovou odměnu
- Distribuce nejsou na začátku známy
- Opakovanými pokusy se můžeme soustředit na nejvýhodnější automaty



Bandit problem

- Exploitation-exploration dilema
  - Dále využívat nejlépe vyhlížející rameno, nebo prozkoumávat méně slibná ramena za účelem získání lepší znalosti o nich?
- Náhodné proměnné $X_{i,n}\ pro\ 1 \leq i \leq K\ a\ n \geq 1$
- μ_i označuje očekávanou odměnu ramene i
- Ztráta po *n* hrách:

$$R_n = \mu^* n - \sum_{j=1}^{K} \mu_j E[T_j(n)]$$

$\mu^* = \max_{1 \leq i \leq K} \mu_i$ je nejlepší možná očekávaná odměna

$T_j(n)$ je počet her ramenem *j* po *n* pokusech

Bandit problem

- UCB – (Upper Confidence Bound)
- Strategie UCB1 [4]: očekávaný logaritmický nárůst ztráty
- Vybíráme rameno *k*, které splňuje výraz

$$k \in \arg\max_{j \in J} \left(\overline{X_j} + \sqrt{\frac{2 \ln n}{T_j(n)}} \right)$$

$\overline{X_j}$ průměrná odměna z ramene *j*

n celkový počet her

Algoritmus

- Každý vrchol stromu reprezentuje stav dané domény
- Orientované hrany vedoucí do potomků představují akce vedoucí do následujících stavů
- Ohodnocení akce odhadnuto na základě **náhodné simulace**
- Postupně zjišťované hodnoty upravují strategii prohledávání
- Iterativní vytváření stromu až do vypršení limitu (čas, paměť,…)

Algoritmus

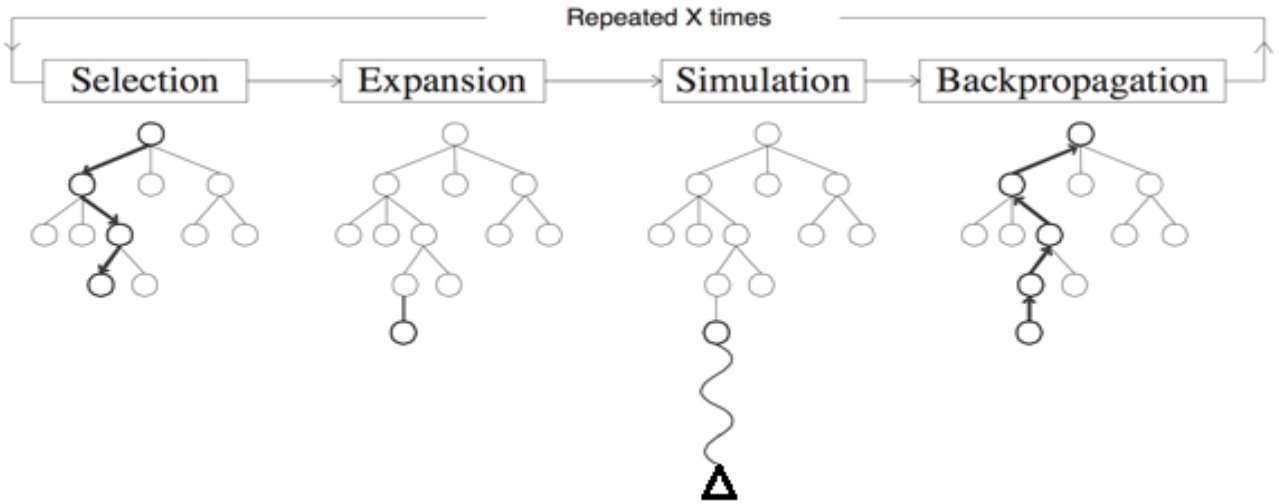
- Každá iterace obsahuje 4 kroky
  - Selekce
  - Expanze
  - Simulace
  - Zpětné šíření

Tree Policy

Default Policy

Algorithm 1 General MCTS approach.

function MCTSSEARCH(s_0)
  create root node v_0 with state s_0
  while within computational budget **do**
    $v_l \leftarrow$ TREEPOLICY(v_0)
    $\Delta \leftarrow$ DEFAULTPOLICY($s(v_l)$)
    BACKUP(v_l, Δ)
  return a(BESTCHILD(v_0))



Algoritmus

- Výběr akce
  - Max child – následník s nejvyšší odměnou
  - Robust child – nejnavštěvovanější potomek
  - Max Robust child – nejnavštěvovanější potomek se současně nejvyšší odměnou
  - Secure child – potomek maximalizující dolní mez spolehlivosti

UCT

- „ UCB + MCTS = UCT”
- Na vrcholy nahlížíme jako na **Bandit Problem**
- Každý potomek vrcholu je nezávislé rameno
- Následník *k* vrcholu je vybrán tak, aby splňoval

$$k \in \arg\max_{j \in J} \left(\overline{X_j} + 2C_p \sqrt{\frac{2 \ln n}{T_j(n)}} \right)$$

Algorithm 2 The UCT algorithm.

function UCTSEARCH(s_0)
  create root node v_0 with state s_0
  while within computational budget **do**
    $v_l \leftarrow$ TREEPOLICY(v_0)
    $\Delta \leftarrow$ DEFAULTPOLICY($s(v_l)$)
    BACKUP(v_l, Δ)
  return a(BESTCHILD($v_0, 0$))

function TREEPOLICY(v)
  while v is nonterminal **do**
    if v not fully expanded **then**
      return EXPAND(v)
    else
      $v \leftarrow$ BESTCHILD(v, Cp)
  return v

function EXPAND(v)
  choose $a \in$ untried actions from $A(s(v))$
  add a new child v' to v
    with $s(v') = f(s(v), a)$
    and $a(v') = a$
  return v'

function BESTCHILD(v, c)
  return $\arg\max_{v' \in \text{children of } v} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 \ln N(v)}{N(v')}}$

function DEFAULTPOLICY(s)
  while s is non-terminal **do**
    choose $a \in A(s)$ uniformly at random
    $s \leftarrow f(s, a)$
  return reward for state s

function BACKUP(v, Δ)
  while v is not null **do**
    $N(v) \leftarrow N(v) + 1$
    $Q(v) \leftarrow Q(v) + \Delta(v, p)$
    $v \leftarrow$ parent of v

UCT

- Nastavováním konstanty C_p lze regulovat poměr explorace a exploitace
- Každý vrchol uchovává 2 hodnoty
  - $T_j(n)$ … počet návštěv
  - $Q(j)$ … součet odměn provedených simulací

$$k \in \arg\max_{j \in J} \left(\overline{X_j} + 2C_p \sqrt{\frac{2 \ln n}{T_j(n)}} \right)$$

  - $\overline{X_j} = \frac{Q(j)}{T_j(n)}$

Charakteristika

- Zaniká potřeba doménově specifických znalostí
  - Stačí umět rozpoznat koncový stav
  - Na rozdíl od hloubkově omezeného minimaxu
- Nicméně přidáním doménově závislých heuristik dosáhneme zlepšení

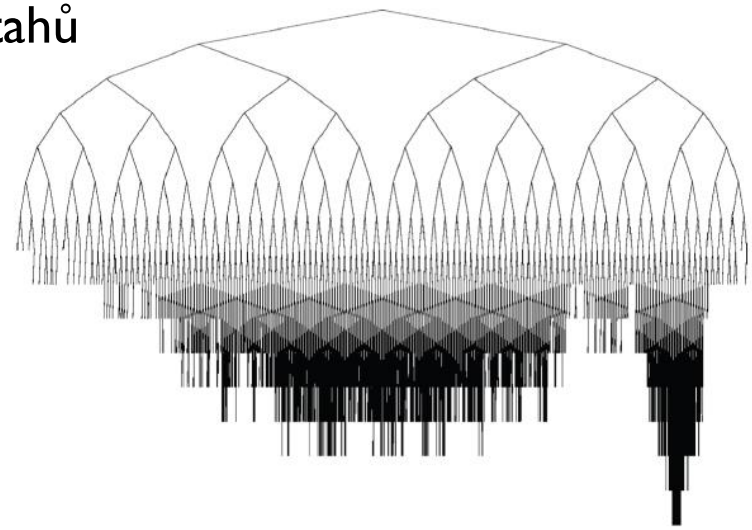
Charakteristika

- Výsledek simulace se propaguje okamžitě
  - Algoritmus se může kdykoli zastavit
  - U minimaxu lze pomocí iterativního prohlubování
- Asymetrický strom
  - Upřednostňování slibných tahů

AHEURISTIC

ANYTIME

ASYMETRIC



Terminologie

Flat Monte Carlo	Náhodně vybírá potomky, nevytváří strom
Flat UCB	Nahlíží na výběr potomků jako na Bandit Problem, nevytváří stom
MCTS	Vytváří strom (používá tree policy)
UCT	MCTS s libovolnou UCB policy
Plain UCT	MCTS využívající UCB1

Vylepšení a heuristiky

Doménově závislé
Doménově nezávislé

- Tree Policy
  - Bandit Based
  - Selekce
  - All Moves as First (AMAF)
  - Prořezávání
- Ostatní
  - Simulace
  - Zpětné šíření
  - Paralelizace

Vylepšení a heuristiky

- Bandit Based:
  - UCB1 – Tuned ,…
- Vylepšení selekce
  - First Play Urgency (FPU)[8]
    - U běžného UCT musí být před expanzí vrcholu rozvinuti všichni jeho sourozenci
    - Problém u velkých větvících faktorů
    - FPU: Ohodnocuje nenavštívené vrcholy pevnou hodnotou, navštívené běžným UCB1

Vylepšení a heuristiky

- Rozhodující tahy
  - Tahy vedoucí k okamžitému vítězství
  - Má-li hráč rozhodující tah, zahraje ho
  - Výhodné, časově náročnější
- Skupiny tahů
  - Redukce větvícího faktoru
  - Shlukování tahů do skupin
  - UCB1 pro určení skupiny, ze které se vybere tah

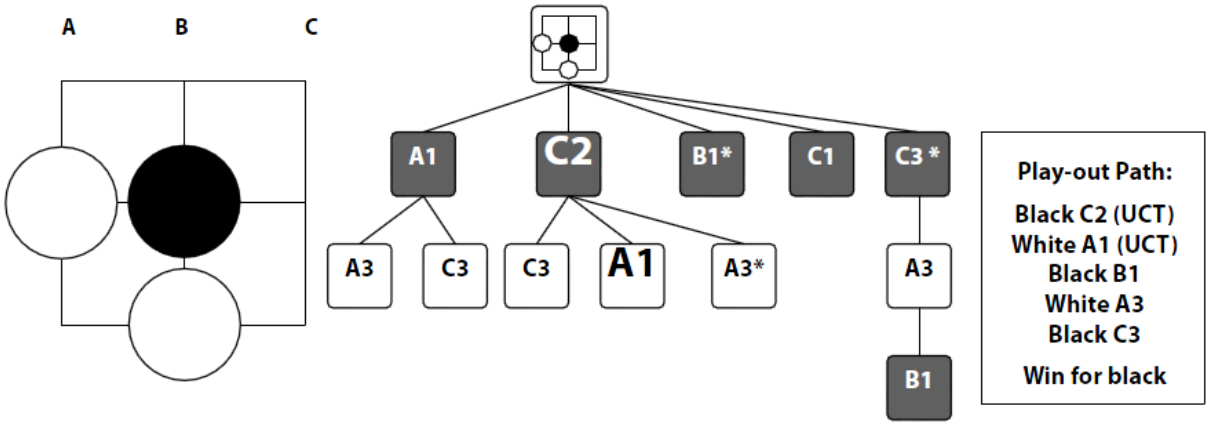
Vylepšení a heuristiky

- Transpozice
  - MCTS vytváří strom, hry mohou být často reprezentovány jako DAG
  - Transpoziční tabulky lze využít ve fázi selekce
- Progressive Bias
  - Přidání doménově specifické heuristická znalosti pro MCTS
  - Málo navštěvované uzly nemají spolehlivě určenou hodnotu
  - Formuli určující výběr následníka rozšíříme o výraz
  - $T_i(n)$ počet návštěv uzlu *i*
  - H_i Heuristická hodnota uzlu *i*

$$\frac{H_i}{T_i(n)+1}$$

Vylepšení a heuristiky

- Databáze zahájení
  - Jsou-li v databázi nalezeny pozice a odpovídající tahy, hrají se
  - Naopak možno využít MCTS pro vytvoření DB
- All Moves As First (AMAF) [9]
  - Basic AMAF
    - Po simulaci aktualizujeme nejen vrcholy, přes které se prošlo, ale i některé jejich sourozence odpovídající danému tahu



Vylepšení a heuristiky

- α-AMAF
  - Sloučení updatu z UCT i AMAF pro každý vrchol
  - α krát AMAF, (1- α) krát UCT
- RAVE (Rapid Action Value Estimation)
  - α pro každý uzel zvlášť
  - Hodnota α se snižuje podle zvoleného parametru *p*: $\alpha_j = \max\left\{0, \frac{p - T_j(n)}{p}\right\}$
- Some First
  - Ze simulované posloupnosti tahů se pro update využije pouze prvních *m* tahů

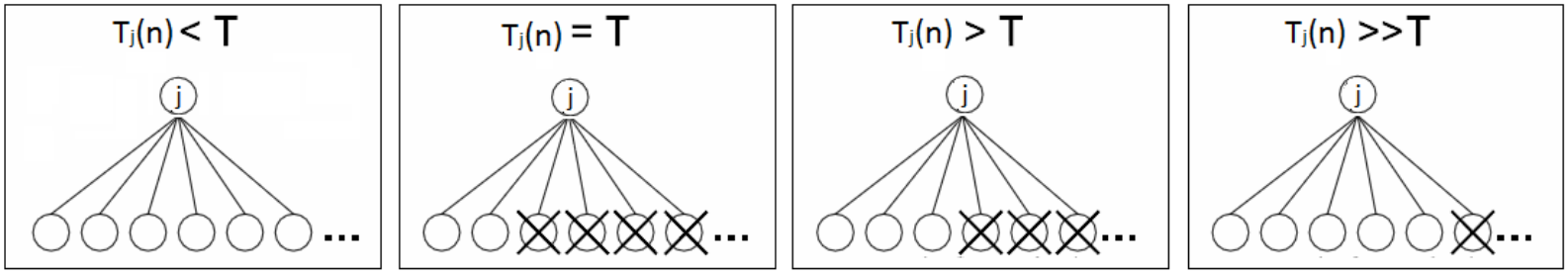
Vylepšení a heuristiky

- Prořezávání

  Soft pruning: tahy mohou být později opět vybrány vs .

  Hard pruning: tahy se nikdy znovu nevyberou

  - Progressive unpruning
    - Dočasné snížení větvícího faktoru
    - U vrcholu, který překročí stanovený práh *T*



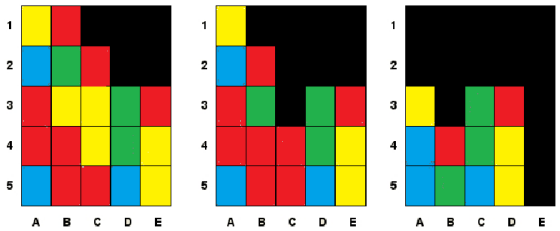
Vylepšení a heuristiky

- Simulace
  - V základní variantě náhodné
  - Možno zahrnout doménovou znalost (online, offline)
- Zpětné šíření
  - Vážení výsledku simulace
    - Pozdější a krátké simulace bývají přesnější
    - Přidání váhy *w* do procesu zpětného šíření

Aplikace

- Hry dvou hráčů s úplnou informací
  - Vynikající výsledky v Go, Hex, (Reversi)
  - U většiny ostatních her (šachy, shogi, arimaa,…) pouze průměrné výsledky

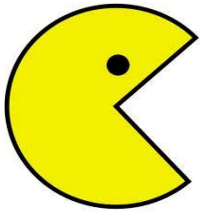
Aplikace



- Hry jednoho hráče s úplnou informací (puzzle)
  - Lloydova 8 (15), Sokoban - tradičně A*, IDA*
  - MCTS varianty úspěšné např. u **SameGame**[5], Sudoku
- General Game Playing
  - Snaha vytvořit agenta, který dokáže hrát více než jednu hru
  - Zajímavé výsledky při použití MCTS metod
  - Mnoho prostoru pro další výzkum

Aplikace

- Real-time MCTS
  - Pac-Man [6]
  - Capture the Flag [7]
  - Množství pokusů na komplexnějších RTS, v kombinaci s dalšími algoritmy



Aplikace

- Nedeterministické hry
  - Skrytá informace a/nebo náhodný prvek
  - Výrazně zvětšuje herní strom
  - Nejlepší počítačový Poker
    - Schopnost nalezení Nashových equilibrií
  - Solitaire

Aplikace

- Ne-herní aplikace
  - Hodnocení zranitelnosti bezpečnostních biometrických systémů
  - Plánovací problémy
  - Optimalizační úlohy (obchodní cestující, hledání nejkratších cest)

Shrnutí

Výhody	Nevýhody
Doménová znalost není nezbytná	Náročné hledání parametrů
Nevhodná znalost výrazně nesnižuje kvalitu	Nevhodné u některých typů stavových prostorů
Bližší lidskému přístupu prohledávání	Špatně předvídatelné chování
Jednoduchost	

- Aktuální téma
- Široké možnosti dalšího výzkumu

Děkuji za pozornost.

Zdroje

- [1] Brügmann: Monte Carlo Go,1993.
- [2] Coquelin, Munos: Bandit Algorithms for Tree Search, 2006
- [3] Arneson, Hayward, Henderson: MoHex Wins Hex Tournament, 2009
- [5] Schadd et Al.: Single-Player Monte-Carlo Tree Search, 2008
- [6] Samothrakis, Robles, Lucas: Fast Approximate Max-n Monte Carlo Tree Search for Ms Pac-Man, 2010
- [7] Chung, Buro, Schaeffer: Monte Carlo Planning in RTS Games, 2005
- [8] Lucas, Browne et Al. A Survey of MCTS, 2012
- [9] All-Moves-As-First Heuristic in Monte-Carlo Go, 2009