Proof-number search, Lambda search a jejich vylepšení

Osnova

- Proof-number search (PNS)
- Úprava na prohledávání do hloubky, PDS a PDS-PN
- Depth-first proof-number search a vylepšení
- Lambda search
- Dual lambda search
- Lambda DFPNS
- Porovnání algoritmů

Použití

- Řešení koncovek her (šachy, Go, dáma)
- Vyřešení her, např. piškvorek
- Situace, kdy jeden hráč tlačí na druhého
- Stromy s různorodým či obecně malým větvením

Opakování: Proof-number search

- Na herní strom se lze dívat jako na strom s AND a OR uzly.
  - Uzly hráče MAX jsou OR (pro dokázání uzlu si stačí vybrat libovolného syna), uzly MIN jsou AND (je třeba prozkoumat všechny soupeřovy varianty).
- Důkaz pozice = výhra hráče MAX, vyvrácení = výhra hráče MIN.
- Uzel je buďto dokázaný, vyvrácený, nebo neznámý.
- Postupně expandujeme strom, vybíráme nejslibnější uzel (Most Promising Node = MPN) na základě proof a disproof numbers.
- Proof-number = minimální počet uzlů potřebných na dokázání uzlu.
- Disproof-number = min. počet uzlů potřebných na vyvrácení uzlu.
- Chová se jako prohledávání do šířky – všechny uzly v paměti.

Vylepšení PNS

- Negamax formulace PNS
  - Mezi úrovněmi prohazuji PN a DN.
  - Všechny uzly tedy počítám stejně jako OR uzly v původním PNS.
- Heuristická inicializace PN a DN expandovaných vrcholů
  - Základní verze používá PN = DN = 1.
  - PN a DN nesmí být nadhodnoceny, jsou to dolní odhady.
  - Odhad obtížnosti důkazu/vyvrácení uzlu.
  - Využití doménových heuristik.
- Obojí lze udělat i pro další algoritmy odvozené od PNS.

Varianty PNS: PN^2

- Dvojúrovňový PNS.
- První úroveň běží v kořeni.
- Pro MPN zvolený v první úrovni se volá PNS pro 2. úroveň s omezenou pamětí.
- Po doběhnutí se celý podstrom vytvořený v 2. úrovni zahodí, až na syny kořene.

Úprava na prohledávání do hloubky

- V PNS po expanzi nebo vyřešení uzlu propagujeme změnu PN a DN nahoru a poté opět vybíráme MPN.
  - MPN ale může být někde hluboko a skončit téměř na stejném místě jako minule.
- Kdy se MPN posune pryč z podstromu uzlu AND?
  - Když PN kořene podstromu přeroste PN nějakého souseda.
  - Analogicky pro uzel OR.
- Odkládáme změnu PN a DN v uzlech nad podstromem.
- Máme tedy prahové hodnoty pro PN a DN, když je překročíme, vrátíme se z uzlu zpět.
- Pozici a její PN a DN uložíme do transpoziční tabulky, jinak celý podstrom zahodíme.
  - Paměťově se chová jako DFS, ale prochází strom stejně jako PNS.

Proof-number and disproof-number search

- Iteruje prahy po jedné PN a DN v kořeni – na začátku 1 a 1.
- Opakované iterativní „prohlubování“ v uzlech stromu.
  - Ve skutečnosti zvětšování prahů.
- Pomalejší než PN^2, když má PN^2 dost paměti
- Předchůdce PDS: PN* – Používá práh pro pouze pro Proof-number
- PDS-PN
  - Rychlejší verze PDS.
  - Běží dvojúrovňově jako PN^2, v druhé úrovni se pouští obyčejný PNS s omezenou pamětí (stejné omezení jako u PN^2).
  - Ukládá prozkoumané uzly v transpoziční tabulce.

Depth-first proof-number search

- Podobný PDS
  - Neiterujeme však v kořeni, ale místo toho nastavíme v kořeni prahy PN a DN na ∞.
  - Opakované iterativní prohlubování ve vnitřních vrcholech.
  - Také používá transpoziční tabulky.
  - Efektivnější než obyčejné PDS.
- Přepočet prahů PN a DN (pro uzel OR):
  - $pt_1 = \min(pt, pn_2 + 1)$, $dt_1 = dt - d + d_1$.
- DFPN+: heuristická inicializace PN a DN listů.

DFPN(r): hry s opakováním

- Graph History Interaction (GHI) problém
  - V některých hrách je stavový prostor graf s cykly.
  - Odlišné cesty do stavu mohou dát různé výsledky.
    - Řešení: zakódovat i cestu (a ukládat ji do TT).
  - Nadhodnocení PN a DN díky opakování.
- Na DAG:
  - Podhodnocení PN a DN.
  - Nadhodnocení PN a DN.

DFPN: 1 + ε trik

- Problém: prahy PN a DN jsou velké a TT nestačí na uložení celého podstromu.
  - Přepočítání podstromu proběhne až lineárně mnoho (vůči PN či DN).
- Úprava vzorce pro výpočet prahů PN a DN v synovi (pro uzel OR):
  - $pt_1 = \min(pt, pn_2(1 + \varepsilon))$ místo $pt_1 = \min(pt, pn_2 + 1)$.
  - Přepočítá jednoho podstromu může proběhnout O(log pt)-krát.
  - Už však nemusíme procházet strom stejně jako PNS.
  - Lze aplikovat i na PDS, zlepšení však není tak velké jako u DFPN.

Lambda search

- Dán binární cíl (mat v šachách, sebrání skupiny v Go).
- Jeden hráč je útočník, druhý obránce.
- λ = řád hrozby: jak rychle může hráč dosáhnout svého cíle.
  - Mat má λ = 0, pro šach λ = 1
  - V Go s cílem sebrat skupinku: λ = počet volností (po tahu).
- Staví se λ^n-stromy, kde n je řád hrozby – postupně od n = 0.
  - Pro n = 0 akorát ověření splnění cíle.
  - Pro n > 0: λ^n-strom se sestává z λ^n-tahů.
  - λ^n-tah útočníka je tah do pozice, pro níž existuje λ^i-strom pro $0 \leq i < n$, pokud obránce vynechá tah (zahraje tzv. null move, resp. pass). Útočník tedy hrozí vyhrát hrozbou nižšího řádu.
  - λ^n-tah obránce je tah do pozice, pro níž neexistuje žádný λ^i-strom pro $0 \leq i < n$. Tj. obránce odvrátil všechny hrozby nižšího řádu.

Lambda search

- λ^{n-1}-strom bývá mnohem menší než λ^{n}-strom.
- Ohodnocení $\lambda^{n}(u)$ pro uzel u:
  - Pro uzel u: $\lambda^{n}(u) = \{ 0, 1 \}$, kde 1 = cíle lze dosáhnout hrozbou řádu n (nebo hrozbou vyššího řádu).
  - Pro list v λ^{n}-stromu: $\lambda^{n}(u) = 1$, když táhl naposledy útočník, jinak $\lambda^{n}(u) = 0$.
  - Pro ostatní uzly Minimax (s α-β prořezáváním), kde MAX je útočník, nebo klidně (DF)PNS.
- Vztahy $\lambda^{n}(u)$:
  - $\lambda^{n}(u) = 1 \Rightarrow \lambda^{i}(u) = 1$ pro všechna i: $n \leq i$.
  - $\lambda^{n}(u) = 0 \Rightarrow \lambda^{i}(u) = 0$ pro všechna i: $0 \leq i < n$.

Lambda search

- Korektnost:
  - Výsledek $\lambda^i(u) = 1$ je korektní, pokud hra má pass nebo zugzwang (nevýhodu tahu) není motivem hry.
  - $\lambda^n(u) = 0$ znamená, že řešení neexistuje, nebo je na vyšším řádu hrozby.
- Algoritmus je dobrý pro problémy, kde řád hrozby má ve hře nějaký význam (např. v Go počet volností). Vhodný hlavně na lokální cíle (někdy i globální).
  - Nepohyblivé „spojovací“ hry: Go, Hex, Havannah, Piškvorky.

Dual lambda search

- V některých hrách může naivní útočný tah hráči uškodit.
  - Druhý hráč má lepší hrozbu – to může λ-search přehlédnout.
- Současně se prohledávají hrozby pro oba hráče.
- μ_p^n-strom se sestává pouze z μ_p^n-tahů (p je hráč, n řád hrozby).
- μ_p^n-tah s následujícími vlastnostmi (o je protivník p):
  - Jestliže táhne p, je to tah do pozice u, ve které, jestliže o zahraje pass (prázdný tah), existuje i: $0 \leq i < n$ & $\mu_p^i(u) = 1$ & $\mu_o^n(u) = 0$.
  - Když táhne o, nesmí existovat i: $0 \leq i < n$ & $\mu_p^i(u) = 1$.
  - Podobné vztahy pro $\mu_p^n(u)$ jako v λ-search.
- Pro ohodnocení vrcholů uvnitř stromu lze použít DFPN.
  - Hodnoty pak budou buď 0, nebo ∞.

Lambda depth-first proof-number search

- Kombinace dvou algoritmů: λ-search a DFPN.
  - PN a DN jsou inicializovány a propagovány jako v DFPN.
  - Uzly útočníka jsou rozděleny na několik pseudouzlů – každý pro jiný řád hrozby (od 0 do l), pro uzly obránce stačí pseudouzly řádů l a l-1 (řád l-1 je kvůli ověření, byl-li předchozí tah útočníka λ^l-tah).
  - Útočník chce dokázat, že $\lambda^l(u) = 1$, obránce $\lambda^l(u) = 0$ (když jsme v uzlu u).
  - Výpočet PN a DN v uzlu u útočníka z pseudouzlů c_1 – c_l:
    - $PN(u) = \min PN(c_i)$, $DN(u) = DN(c_l)$.
    - Analogicky pro uzly obránce.

Porovnání algoritmů

- Z experimentů v článcích na hrách Go, Shogi, Lines of Actions …
  - Většinou koncovky, v Go řešení jedné nezávislé části pozice.
  - Téměř vše je lepší než standardní α-β (příp. i s nějakým rozšířením).
  - Obyčejný PNS je příliš náročný na paměť.
  - PN^2 je 3krát rychlejší, než PDS, ale vyžaduje více paměti.
  - PDS-PN > PN^2 – obzvláště s omezenou pamětí (Lines of Actions).
  - DFPN 1+ε > DFPN > PDS 1+ε > PDS (Lines of Actions).
  - DFPN vs. PDS-PN? (Dnes se používá více DFPN.)
  - λ-search vs. DFPNS?
  - (DF)PNS oproti λ-search nemá problém s zugzwangem.
  - Dual λ-search > λ-search (Shogi).
  - LDFPN > DFPN+ (Go).

Reference

- Winands H., Uiterwijk J., Herik J.: *PDS-PN: A New Proof-Number Search Algorithm*
- Pawlewicz J., Lew L.: *Improving Depth-rst PN-Search: 1 + ε Trick*
- Kishimoto A.: *Dealing with Infinite Loops, Underestimation, and Overestimation of Depth-First Proof-Number Search*
- Thomsen T.: *Lambda-Search in Game Trees – with Application to Go*
- Soeda S., Kaneko T., Tanaka T.: *Dual Lambda Search and Shogi Endgames*
- Yoshizoe K., Kishimoto A., Müller M.: *Lambda Depth-first Proof Number Search and its Application to Go*

Závěr

- Dotazy
- Další varianty PNS a články o nich.
  - Focused DFPN – pro širší stromy
  - Evaluation-function based PNS
  - Paranoid PNS – pro hry více hráčů
  - Weak PNS – odstraňuje problémy s DAG
  - Job level PNS, Parallel DFPN – paralelizace (DF)PNS
  - Kombinace PNS a MCTS
- Diskuse
  - Kombinace Dual lambda search a DFPN?
  - Kombinace (Dual) Lambda search a MCTS?