Herní algoritmy, předn. 2,3

Jan Hric, Petr Baudiš

Minimax

NAIL103, ZS

8. listopadu 2012

Obsah

Minimax - z minula

- Základy: Minimax, alfa-beta, ...
- Varianty: Negamax, Principiální varianta, (Negascout), Iterativní prohlubování, MTD(f), (SSS*) ...
- Impl: Transpoziční tabulky, okno, history heuristika, killery, null-move, ...
- Související pojmy: heuristická ohodnocovací funkce, efekt horizontu, ...

Terminologie, základy

- kontext: deterministické hry dvou hráčů s úplnou informací a nulovým součtem (a střídáním tahů)
- pozice, prostor stavů, strom/graf hry, tah, historie tahů
- Stav hry: úplný popis, (kdo je na tahu, ”rošáda”), (část) historie
- Výsledky: boolovské $\{0, 1\}$, $\{-1, 0, +1\}$; numerické, často $(-N, +N)$
- reprezentace: stromy, DAG, (DCG (cyklické)) - konečné
- And/Or strom, strom důkazu (resp. vyvrácení): popis vyhrávající (resp. prohrávající) strategie (dis/proof tree)
- - strategie: v každém vrcholu určí (nejlepší) tah, pouze některé vrcholy jsou relevantní

Minimax

- Boolovské hledání: vrací 0 nebo 1
- koncové vrcholy: staticky, Max vrchol: or, Min-vrchol: and (And/Or stromy se používají i v jiných apl.)
- Negamax: z pohledu hráče na tahu; impl.: vrací opačnou hodnotu
- Hodnoty z intervalu: operace min a max; používáme jemnější rozdělení pozic, pokud nedopočítame hru do konce
- Redukce na bool. případ se používá v Negascout/PVS
- Pokud $v \geq m$ je výhra a $v > m$ je prohra, pak m je minimax. hodnota

Alfabeta

- udržuje dolní a horní odhad (alfa,beta) skutečné hodnoty
- (tj. verze/analogie: branch and bound)
- pokud je hodnota v mimo interval, usekni další syny v
- když $v < alfa$, my do v nepůjdeme, máme lepší možnost
- když $v > beta$, soupeř do v nepůjde, má lepší možnost

Alfabeta

- potřebuje dobré uspořádání tahů
- nejvíc ušetří, pokud je první vrchol nejlepší: dovolí jít do dvojnásobné hloubky
- alfabeta vrací přesnou hodnotu *m* minimaxu, proto musí prohledat aspoň proof tree a disproof tree

Strom důkazu

Jak vypadá (výsledný) strom důkazu a strom vyvrácení?

- strom důkazu v Max vrcholech (včetně kořene) obsahuje 1 syna, v Min vrcholech všechny syny. Koncové pozice jsou vyhrané (z hlediska kořene).
- strom vyvrácení v Max vrcholech obsahuje všechny syny, v Min vrcholech 1 syna. Koncové pozice jsou prohrané.
- průnik stromů je jedna cesta - principiální varianta: oba hráči hrají (neostře) nejlepší tahy
- pro regulární strom se štěpením b: oba stromy se štěpí plně na každé druhé úrovni, tj. efektivní štěpení je $\sqrt{b}$
- - důsl: v nejlepším případě Alfabeta prohledá *za stejný čas* v porovnání s Minimaxem strom do dvojnásobné hloubky

Heuristická fce

- zatím prohledáváme do konce hry - pro přesný solver
- reálně: heuristicky odhadneme hodnotu, po *d* tazích, pomocí *statické, ohodnocovací, heuristické funkce*
- prohledávání do omezené hloubky se použije pro uspořádání tahů:
  - potřebujeme transpoziční tabulky, protože prohledáváme opakovaně
  - používáme ID: iterační prohlubování (iterative deepening), s oknem
- chceme rychlost: prohledání do hloubky o 1 větší s horší funkcí je lepší (obvykle)
- vytvoření dobré (a rychlé) heur. fce je samostatná otázka, doménově závislá
- různé přístupy k tvorbě: ručně/expert (nejlepší), (samo)hraní, učení z pozic a her expertů, (testovací sady)

Heuristická fce: impl.

- pozorování: na konkrétních hodnotách heur. fce nezáleží, pouze na uspořádání; (v kořeni vybíráme 1 nejlepší tah)
- tj. ”lepší” pozice (např. s větší pravd. výhry) má mít vyšší hodnotu heur. fce
- typicky vyjadřuje (šachy:) kvalitu/převahu materiálu, (hex:) možnosti spojení a blízkost k cíli, ...
- jednoduchá impl: vážená lin. funkce rysů (features), důležitější rysy mají větší váhu, (tvar používaný při učení heur. fce, protože je jednoduchý a lze dobře učit); ale: všechny rysy musím počítat, v různých druzích pozic jsou důl. různé druhy rysů
- rys: vlastnost pozice, např. umístění kamenů, souhra (tj. patterns), ... (Q: jak zjistit druhy vzorů?)
- inkrementální počítání hodnot, (postupné počítání - ”jemné” (málo významné) rysy počítáme pouze blízko správné v, ...), diskrétní hodnoty: $1/100$ pěšce ($=$ 1 centipawn)

Problém horizontu

- statické ohodnocení funguje (v šachu) spolehlivě v tzv. tichých pozicích
- v pozicích takticky nestabilních (braní, šach, výměna figur) je vhodné pokračovat v prohledávání (některých tahů) → tiché prohledávání (př: ohodnocení uprostřed výměny)
- Problém horizontu: v situaci, kdy soupeř má vynucující tah se zlými následky, ale my můžeme vynutit odložení tahu za aktuální hloubku (horizont).
- Pak náš vynucující tah má lepší ohodnocení, i když nic nevyřešil (a ještě hůř: případně způsobil dodatečnou nevynucenou "malou" ztrátu → strategické zhoršování pozice),
  - př: Arimaa: mělké hledání, ostré pozice

Problém horizontu 2

- není známo univerzální řešení
- i v MCTS podobné chování způsobuje nestabilitu: místo řešení těžké/nevýhodné situace engine hraje (v go) vynucovací tahy, řešení odkládá za horizont, do playoutů
- je těžké odlišit P.h. od *vhodného* vynucovacího tahu (go: forcing move, zkouška odpovědi)
- př: go: podle toho, zda schody fungují, si několik tahů dopředu vybírám/zavrhuji odpověď (typicky v joseki - zahájení)

Alg. založené na alfabeta

- Idea: menší okno způsobí víc ořezání
- *Null window* – ekvivalentní boolovskému hledání
- S dobrým uspořádáním tahů, hodnota prvního tahu usekne ostatní tahy
- Změna strategie: riskantní (”optimistická”), ale správná, protože znovuprohledává, pokud je to nutné
- alg. Scout, Negascout: použije nulové okno pro useknutí tahů kromě prvního
  - lepší možnosti paralelizace
- PVS - Principal Variation Search (ekviv. Negascout)
- ad: mnoho metod pochází z šachu (”pilotní” projekt, šach. intuice, (přímé) komerční apl.)

Zmenšení okna

- Klasické okno: (-inf, +inf)
- Useknutí může nastat až po dokončení prvního tahu
- Pokud máme ”dobrý odhad”: např. ”aspiration window” v šachu: skóre z minulého tahu (nebo minulého mého), $\pm$ hodnota pěšce
- riskantní: pomůže a ušetří nebo selže a přidá (”selže v nestabilních situacích”)

Chování okna

voláme alfabeta s konečnými alfa a beta
Možné výsledky, hodnota je v

- $v \in (alfa, beta)$: v je správná minimax. hodnota
- $v \leq alfa$: tzv. fail-low - prohledej s oknem (-inf, alfa)
- $v \geq beta$: tzv. fail-high - prohledej s oknem (beta, +inf)

Implementačně: ! transpoziční tabulky; failsoft: (bez dodatečné práce) vracení hodnoty i mimo interval:

- vracím hodnotu v mimo (a, b)
- pokud $v <= a$, potom je to nová horní hranice (beta), protože všechny uzly jsem (částečně) prohledal
- ... a symetricky: $v >= b$, mám novou hodn. alfa

Nulové okno

- volání alfabeta s beta = alfa + 1
- chová se jako boolovské hledání s testem $v < beta$

Principiální varianta

- PV je sekvence, kde obě strany hrají nejlepší tah
- Vrcholy na PV mají stejné hodnoty jako kořen
- Žádný hráč nedokáže zlepšit tahy nad hodnotu PV
- (Může být víc PV variant; ...)
- Obvykle se PV nazývá první objevená varianta, ostatní jsou useknuty; s nulovým oknem máme pro další tahy pouze jednostranný odhad

PVS/Negascout

- Idea: první tah prohledáme a určíme dolní mez v
- Boolovské (tj. null window) hledání ověří, že ostatní tahy mají hodnoty $\leq v$
- Při neúspěchu (fail-high) znovuprohledáme a určíme novou *lepší* hodnotu
  - varianta: speciálně (pouze) v kořeni: znovuprohledání až po druhém fail-high
- Spoléháme na dobré uspořádání tahů (...), pak je znovuprohledávání řídké a celkově se vyplatí

Vylepšení prohledávání

- Základní alfabeta alg. je jednoduchý, ale nedokonalý
- Pro výkonný herní engine protřebujeme mnoho rozšíření
- Obecné (nezávislé na hře, na algoritmu), doménové
- I mnohé ”obecné” fungují jen v některých hrách (...)
  - Hry jsou různého druhu, rozšíření využívá implicitně nějakou charakteristiku hry
  - Př: ”spojovací hry”: hex, go, piškvorky, (tzaar)
- Projdeme některé nejdůležitější rozšíření

Souvislosti, k zamyšlení ...

- je mnoho variant algoritmů a rozšíření, s mnoha parametry
- chování závisí na mnohém: velikost a struktura stromu, dostupnost doménových znalostí, tradeoff rychlost - kvalita, sekvenční vs. paralelní
- často citlivé na parametry, na implementaci, na kombinaci technik (synergie, předpoklady)
- závislé na doméně (na hře)
- kvalitní alg. je pracný, měření, (např. závislost parametrů na hracím čase, na fázi hry)
- přínos vs. režie, (nedokonalá impl., která vytěží - v průměrném případě - podstatu zlepšení: např. TT, fáze hry podle materiálu nebo č. tahu)
- (impl. komerčních programů jsou neveřejné)

Druhy rozšíření

- Přesné (vrací správnou minimax. v) vs. nepřesné (heuristické)
- Zlepšování pořadí tahů
- Zlepšování chování prohledávání
- Zlepšování paměťového chování (pruning)

Nezávislé na hře i specifické pro hru

Iterativní prohlubování, ID

- Idea: postupné prohledávání s hloubkou $d = (0), 1, 2, ...$
- Výhody:
  - Anytime alg. - lze přerušit, na začátku je rychlý
  - pro b velké je malá režie, poslední iterace dominuje
  - s transpozičními tabulkami lze ukládat nejlepší tah pro zlepšení uspořádání tahů
  - často prakticky rychlejší
- některé programy/hry inkrementují d po dvou (fluktuace sudá/lichá při ohodnocování, malé b)

ID a řízení času

- při pevném/tvrdém časovém limitu nutný abort poslední iterace
- uschováme nejlepší tah z poslední dokončené iterace
- zkusíme predikovat, zda dokončíme další iteraci
- lze použít neúplnou poslední iteraci, pokud se stihl dopočítat aspoň jeden tah (ale první tah je zdaleka nejpomalejší)
- (přidávání času při nestabilitě výsledků mezi úrovněmi: něco se děje na horizontu)
- použití Opening book nechá víc času pro střední hru (a lepší pozici)

Transpoziční tabulka

- je hašovací tabulka, ukládá pozice, poskytuje *perzistenci*: sdílím(!) info mezi tahy, mezi úrovněmi ID
- výhody: nemusím prohledávat stejný podstrom dvakrát, např. když stejná pozice vzniká přehozením tahů; TT si pamatuje data explicitně, např. nejlepší tah z předchozího prohledávání
- podstatné v DAG, kde je víc cest k vrcholu
- nevýhody: GHI problém (Graph History Interaction); zvyšuje nestabilitu: např. vrací hlubší hodnoty, nezaznamenává cestu k pozici (!úmyslně)
- (měření: i malé tabulky pomáhají, zvětšování pomáhá ”trochu”)
- impl: inspekce přítomných synů před dalším prohledáváním (Enhanced Tranposition Cutoff, ETC); pozorování: i když položka vypadne z TT, (někteří) synové zůstanou
- Otázka designu: ukládat statické ohodnocení, (tiché dohledávání) - závisí na rychlosti, velikosti tabulek ...

Transpoziční tabulka: obsahuje

- klíč pozice pomocí Zobristova hašování: rysy popisu pozice se zobrazí na čísla a Xor-ují
  - ! kóduje *úplnou* informaci o pozici: kdo je na tahu, šach: možnost rošád(y), go: zákaz ko
- obvykle není 1:1; někdy druhotný klíč pro odstranění kolizí
- hodnotu pozice; (doporučeno u MTD(f) (využívá null window, ...): samostatné položky pro dolní a horní odhad)
- flagy: přesná hodnota, dolní mez, horní mez
- hloubku prohledávání, případné další nastavení enginu
- nejlepší tah

Transpoziční tabulka, impl.

- lze komprimovat (flagy, hloubku, hodn) do 1 slova
- často nevhodné, ale šetří prostor a umožní větší TT
- velikost TT typicky mocnina 2, aby se 2^n vešlo do paměti
- prvních n bitů je index do TT
- lookup: kontrola celého hashe (nebo druhotní klíč); ! chyba, pokud stejný hash kód má jiná pozice (...)
- uložení: strategie řešení konfliktů - pouze se přepisuje
- různé strategie přepisování; (oblíbené:) v položce 2 sloty: 1. poslední, 2. největší/nejhlubší, resp. nejlepší

Zobristovo hašování

Rychlá heuristická metoda pro počítání hash kódu. Idea:

- Jednoznačný náhodný hash kód pro každou dvojici (pole,hodnota), a pro další informace o pozici
- - obvykle používá 0 pro jednu hodnotu (prázdné pole)
- hash kód pozice je Xor kódů polí
- změna: jeden Xor pro změněné pole (tabulka změn mezi figurami), (opět) počítá se inkrementálně, (tabulky včetně braní, změny figury, ...)
- Undo tahu: taky Xor (go, při braní skupiny: pamatuju si rozdíl)
- typicky aspoň 64 bitů pro kód
- heuristické kódy lze použít i pro solver, ale musíme *verifikovat* důkazový strom
- jiná možnost: přesné uložení pozice, (velké s historií), hex/go: n polí, 3 druhy obsazení: $\lceil n \log_2 3 \rceil \approx 1.58n$ bitů

Uspořádání tahů

- je důležité
- iterativní prohlubování je efektivní, používá paměť v TT
- možnosti: best move z TT, killery, history heuristic, statické ohodnocení, přizpůsobený generátor tahů (– vhodně promíchané, záleží hlavně na prvních tazích) - dobrý tah na začátku aspoň omezí okno (pozorování: většina tahů je špatných/nerelevantních)
- často specifické heuristiky pro konkrétní hru, př. hrozba matu; preferování určitých druhů tahů v generátoru
- obecnější: specifická ohodnocovací funkce
- př: uspořádání synů podle (statické) ohodnocovací funkce

Heuristika historie

- History heuristic: zlepšuje pořadí bez herně-specifických informací
- dává bonus tahům, které způsobily useknutí (cutoff), často podle velikosti ořezu (n^2 nebo 2^n pro výšku n)
- preferuje tyto tahy na jiných místech při prohledávání (!když jdou použít); vlastně využívá korelaci hodnot/tahů
- impl: index je (”from”,”to”) nebo (”figura”,”to”)
- (podobnost s AMAF/RAVE v MCTS: All Moves As First/Rapid Action Value Estimation), obecně: snaha získat z prohledávání/playoutů další užitečné informace (sdílení informací)

Killery

- killer move: použiju nejlepší tah (tj. způsobující cutoff) ze stejné úrovně (tj. z bratra) jako první/na začátku
- impl: pro úroveň si pamatuju (např. 2) killery; lze je zkoušet *před* generováním tahů; (impl.: killery nahrazuju např. podle hodnoty (posledního) tahu)

Úpravy hloubky

V základní variantě: do pevné hloubky *d*
Idea: slibné tahy hlouběji, málo slibné redukovaně hluboko
Přesné i heuristické metody, (silně) závisí na hře: mnoho šachových rozšíření
Možnosti úprav:

- rozšíření (extensions) - zvýšení hloubky: ! tiché prohledávání (quiescence search); průběžně vs. v listech ...
- redukce - snížení hloubky (jednorázově nebo rekurzivně): zlé statické ohodnocení (př. dáma jako poziční hra: při ztrátě materiálu v aktuální větvi), razoring (ve výšce 2 (nad listy) useknu vrcholy se statickým val$\leq$alfa; má varianty)
- useknutí - ale heuristické: null move heuristic; podle statického ohodnocení - může způsobit nestabilitu nebo překvapení

Useknutí může vést k přehlédnutí taktické varianty nebo přehlédnutí (mého) dobrého nestandardního tahu. Redukce hloubky je bezpečnější.

Příklady úprav

- Nulový tah, Null move
- ProbCut
- Zlomková hloubka
- Tiché dohledávání, Quiescence search
- Late Move Reduction (LMR)

Nulový tah

- Pozorování: skoro všechny větve mají aspoň jeden špatný tah
- Idea: useknout tyto podstromy dříve
- Předpoklad: ve hře každý tah zlepšuje pozici
- Null move: pokud pasujeme a plytší prohledání stejně způsobí ořez (fail low), pak usekni podstrom
- (soupeř ani s tempem/dvěma tahy nemá dobrý výsledek)
- problém: zugzwang (obojstranná nevýhoda tempa: kdo táhne, prohraje), šach: málo častý vs. dáma: častý; go: (pouze lokálně, v go je *pas*) seki - stav zůstane v koncové pozici/"nedohraný"
- (odhaluje hrozby proti, obecně nestabilní situace), podobné s λ-prohledáváním

ProbCut

- Pozorování: v mnoha hrách jsou (s dobrou ohodn. fcí) výsledky prohledávání vysoce korelované mezi úrovněmi
- Idea: redukovat hloubku pro tahy, které jsou pravděpodobně špatné
- (má varianty a parametry; pokud hledání s reduk. hloubkou vrátí hodnotu (hodně) mimo okno, věřím ji, jinak znovuprohledám; v pevné výšce nad listy)
- ponechá víc času pro slibné varianty

Zlomková hloubka

- Idea: rozšířit hledání po vynucovacích (forcing) tazích (a jiných slibných druzích tahů)
- Impl: redukovat hloubku o méně než 1
- Př: šachy: hrozba šachu a únik, hrozba braní; go: hrozba braní (atari) a únik: $1/8$ ceny
- Chování: větve s mnoha vynucenými tahy jsou prohledány mnohem hlouběji
- nebezpečí: velikost stromu se může (prudce) zvýšit
- Redukce prohledávání: zmenšíme hloubku o víc než 1 pro méně slibné větve

Tiché dohledávání

- Pozorování: je těžké ohodnotit nestabilní pozici
- př. šachy: šach, uprostřed výměny
- Idea: ohodnocujeme pouze ”stabilní” pozice
- v nestabilních pozicích nahradíme statické ohodnocení omezeným *tichým dohledáváním*
- používá *velmi omezený* generátor tahů - chceme pouze odstranění nestability

Late Move Reductions

- LMR, ”redukce zbylých tahů”, podobné k: prořezání podle historie (history pruning), redukce podle historie (history reduction)
- Idea: v pravděpodobných fail-low uzlech, redukujeme prohledávací hloubku níže zařazených tahů
- nepoužívá se na taktické tahy
- důsledek: tvaruje strom

Paměťové a časové nároky

- Čas: makeMove/undo a logika alfabety jsou rychlé; heuristická funkce taky
- mnoho info lze počítat inkrementálně (po odladění)
- Paměť: alfabeta (a varianty) je prohledávání do hloubky, proto má velmi malé paměťové nároky
- - pro porovnání SSS* je prohledávání do šířky
- Dodatečnou paměť lze použít pro transpoziční tabulky

Realization Probability Search

- Možnost, jak systematicky nastavit zlomkovou hloubku (pro kategorie tahů)
- Idea: zavedou se kategorie tahů, každá má svou zlomkovou hloubku
- Realization Probability Search, Enhanced Realization Probability Search
- Odhadne se pravděpodobnost, že další tah je v určité kategorii ze záznamu partií mistrů, podle toho zlomek hloubky
- - (na podobných principech druhů tahů je založeno učení ohodnocovacích funkcí, tahy mají rysy a hledáme vhodnou kombinací rysů tahu pro určitou pozici, resp. typ pozice)

Enhanced Transposition Cutoffs, ETC

- Idea: některé syny lze najít v TT, možná způsobí ořez
- Tradeoff: mezi prací navíc a úsporou. Možná vyzkoušet jen pár synů? Možná použít jen dost vysoko (nad listy)?
- (drahé akce používáme pouze pro nejslibnější vrcholy anebo pouze vysoko, v MCTS tzn. často navštívené - počet nad prahem)
- Nekompatibilní s kešováním
- (Plaat a kol. 1996)

Další algoritmy

jiný princip, jiné procházení stromu: MTD(f), (SSS*)

- Dosud PVS, Negascout: plné prohledání prvního tahu (modulo ID), null window pro ostatní tahy
- MTD(f): Memory-enhanced Test Driver with node n and value f: dělení intervalu nulovým oknem
  - MTD(f): pouze null window hledání
  - Idea: upřesňuje hodnotu pomocí dělení intervalu - tj. okna
  - Možný postup: binární dělení; hodnota z minulého tahu/ID a posun nahoru nebo dolu; příp. malý bonus ve směru změny
  - Lze použít failsoft, znovuprohledání podobné Negascout, ale častější
  - závisí na jemnosti heuristické funkce, dobrém prvním odhadu
  - Doporučeno: v TT mít dvě položky pro horní a dolní odhad zvlášť

Další algoritmy 2

- (SSS*: State Space Search) - prohledávání do šířky podobné A*, proof a disprof tree *v paměti* a rozvíjím nejslibnější uzel
  - taky obsahuje exploration-exploitation dilema
  - podobný princip použitelný pro budování Opening book
  - překonaný MTD(*f*)

... a další poznámky

- turnaj polotahů, chytrá vs. jednoduchá heur. fce, pravidlo klesající návratnosti (dimmishing returns): víc práce/režie a méně užitku (vylepšení jsou částečně nahraditelná)
- refutační tabulky (dříve místo TT, dnes opět v playoutech v MCTS): pamatuju si slibné odpovědi na tahy
- v soutěžích (vs. experimenty): *pondering* - počítání na čas soupeře
- opening book: experti se neshodnou, program v omezeném čase nedokáže dobře využít/vyřešit získanou pozici; (dnes) často automaticky budované a upravované
- (endgame tabulky)

... a další poznámky 2

- druhy vrcholů (původně Knuth): PV, Cut node (fail-high, beta cutoff), All node (fail-low)
- analýza plauzibility, tvrdé prořezávání tahů (např. šógi)
- (nesymetrické prořezávání: soupeře prořezávám míň, nechci přehlédnout taktickou variantu vs. u sebe přehlédnu nestandardní tah v konkrétní pozici)
- (množství technik: nepřehledné, i drobné varianty mají ”jména”: alg. se musí odlišit a mít přínos (aspoň na svých testovacích datech :-)); ale nové varianty odstraňují problémy nebo nevýhody starých - někdy ”drobnou” změnou; programy jsou ”nerobustní”; techniky jsou nezávisle implementovatelné a vzájemně kombinovatelné (někdy se synergií))
- impl: nestabilita v prohledávání, nutno ošetřit (je důsledkem prořezávání), př.: po fail-high následuje fail-low (ve správném okně, ”za jinak stejných podmínek”)

Nestabilita ohodnocovací funkce

program se chová (z nadhledu) nerozumně jako důsledek nedokonalosti heur. fce v některých situacích; synergie rysů

- Bylo: Problém horizontu
- Problém: potenciál zahrání tahu je cennější než vlastní zahrání tahu, příp. kombinace (arimaa: hrozba braní, taky zajetí/hostage)
- Problém: cesta k realizaci hrozby vyžaduje ”tunelový” efekt (arimaa: dvojtahové braní; a malá hloubka prohledávání) - mezilehlé pozice jsou méně vhodné; př: arimaa: poziční ohodnocení malých figur je velké u krajů a je těžké je přemístit
- $\rightarrow$ nutné ladění funkce (příp. automatické)
- MCTS: shallow traps; go: nakade - tahy dovnitř (pro zabití skupiny)

Různé druhy her

- různé druhy her, jejich charakteristiky, jejich cíle: spojovací, přirozeně plynoucí (Othello/Reversi), s nepohyblivými figurami, poziční, materiálové, akumulační(body)/zajímací (Oware), omezovací (soupeř nemá tah), náhlá smrt a víc cílů (i obranných), shromažďovací (souhra, lokální přesila: Abalone, TZAAR), ...
  - charakteristiky a cíle nejsou nutně stejné; charakteristiky jen ”do určité míry”
  - různé chování v různých fázích hry (nutno vyřešit určení fáze hry)
- příklady her: Lines of Action (spojování), Havannah ($\approx$ Hex, víc cílů), Quoridor (poziční), Abalone (přetlačování, zajímání, kontaktní hra), Amazons (omezování), Gobblet (přikrývání) ...
- GIPF: posouvání a braní, ZERTZ: obětování, DVONN: akumulace a věže, YINSH: obracení, PÜNCT: spojování, TZAAR: věže a zajímání, 2 stepy, (TAMSK: čas)

Co je těžké/nevhodné pro počítač

Jak navrhnout hru, aby byla těžká pro počítač:

- široký strom (Arimaa, Amazons), př: víc stepů; šógi: pokládání zajatých figur (branching 80-120, i 500)
- těžká ohodnocovací funkce (viz dále)
  - náhlá smrt (a forced moves): to využije lambda search a PNS, ale ty jsou pomalé pro engine
- několik lokálních ”bitev” (go - taky *tenuki*: přerušení lokální sekvence; Arimaa, Havannah), několik možností výhry
- variabilní poč. pozice: Opening book šetří čas pro střední hru a zabraňuje chybám;
- znemožnění tabulky koncovek
- ”strategické pojmy”; př. arimaa: sbírání materiálu (postupně; i 1 figura na víc tahů)
- (spíš pro solver:) závislost na historii (např. zákaz třetího opakování), GHI → nestabilita TT

Těžká ohodnocovací funkce:

- pozice je citlivá na malé změny, tj. podobné pozice mají velký rozptyl hodnot (a e.f. neví, kam se dívat)
  - pozice jsou (často) ostré: Arimaa
  - je těžké uspořádat tahy: 1-ply nelze dobře použít, nestabilní ID
  - :-) rozvalinové go, ale není to typický průběh hry
- ”dlouhé” figury: vše souvisí se vším - aspoň občas; go: schody, Stratego: scout, (e.f. nemůže vyloučit důsledky vzdálených figur)
- je těžké najít správný protitah soupeře; (shallow traps: původně MCTS)
- rysy eval. fce jsou kombinované, v různých druzích pozic jsou důležité jiné rysy
- ”různé” eval. fce pro různé fáze hry (jednoduchý př: poker: počet hráčů u stolu, ...)

Učení eval.f.

úspěšný příklad (strojového učení):

- ad: učení e.f.: Bonanza method, Kunihito Hoki, 2008, zveřejněný kód
- Šógi, pgm. Bonanza, vítěz WCSC 2006, napoprvé
- koef. pro všechny dvojice figur na 9x9 (asi 10^6)
- aproximovat tahy mistrů

komentář: použití hrubé síly jiným způsobem (než minimax)

Zdroje

- S. Russel, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 2003 (kap. 5)
- Martin Müller, přednáška na University of Alberta: `http://webdocs.cs.ualberta.ca/~mmueller/` (asi nedostupná)
- `chessprogramming.wikispaces.com/`

Různé

- Doménové znalosti, odstraňování nevýhod (go: nakade, criticality; Arimaa: zajetí velblouda)
- MCTS pro jiné aplikace: SP ..., hry s náhodou, skrytou informací
- Učení heuristických ohodnocovacích funkcí (Backgammon: Tesauro 1992, reinforcement learning), self-play, ladění parametrů (velký prostor, velký rozptyl, citlivé)
- Učení Opening book, analýza pozice
- Hry těžké pro počítače: Arimaa, (Game of the) Amazons, GIPF project (Dvonn, Gipf, Zertz, Yinsh, Punct, Tzaar, (Tamsk))
- SAT, QBF: Quantified Boolean Formula - And/Or strom
- CGT: Combinatorial Game Theory (viz)
- porovnávání kvality (nejen šachy): body Elo, lepší hráč má vyšší pravděp. výhry; (kvantitativní model $\rightarrow$) pravidla úpravy podle Elo hráče a rozdílu soupeřů při výhře/prohře, (jiné modely: Glicko, TrueSkill)

Možné témy referátů

Základní techniky jsme spíš probrali
- další varianty PNS, (μ-search)
- hry s neúplnou informací a náhodou (Stratego, poker)
- hry 1 hráče, Single player MCTS (SP-MCTS)
- hry více hráčů
- učení heur. fce, často self-play, reinforcement learning, př: backgammon
- GGP: General Game Playing (možná popis některého systému: CadiaPlayer, Ary, FluxPlayer, ...)

Témy spíš k diskusi (malé/speciální/nevhodné)
- budování Opening book, (tabulky koncovek)
- ELO rating a jiné (Glicko, TrueSkill), (Bradley-Terry model, ...)
- (GHI), (učení vzorů, doménových znalostí), (paralelizmus)

Enhanced Realization Probability Search

ERPS: M. Winands 2007, (RPS: Tsuruoka 2002)

- Pozorování: Různé druhy tahů se hrají s různou Pstí. (My chceme simulovat experty.)
- Tahy rozdělíme do disjunktních kategorií. Kategorie mají Pst zahrání, které určíme z tahů expertů. (pro dostatek dat: ”Expert je libovolný lepší hráč”)
- Pst kořene je 1, Psti po cestě se násobí. Při podtečení prahu t se větev ukončí.
- Impl: Psti transformujeme na zlomkovou hloubku $FP = \log(P_c)/\log(C)$, kde $C, 0 < C < 1$ je konst. závislá na hře, $P_c = n_{zahrany(c)}/n_{hratelny(c)}$ je poměr počtu (pozicí se) skutečných zahrání k možným zahráním (podle pravidel hry), pro tahy z kategorie c
- předcházení problému horizontu: alg. *hlouběji* znovuprohledává vrcholy, kterých hodnota je větší než aktuální best value, s použitím malé změny hloubky: *minFP*

ERPS, zlepšení

- dvě vylepšení v ERPS:
- 1. tahy s fail-low jsou vždy ignorovány
- - prořezávání se aktivuje až po *m* tazích (podle pořadí)
- - prvních *m* tahů se hloubka redukuje s *minFP*
- - analogie s LMR a RankCut
- 2. tahy s fail-high jsou znovuprohledávany s *minFP* (i slabé tahy)
- - vřadí se znovuprohledání s hloubkou průměru mezi původním FP a minFP
- - jen pokud redukované prohledání opět skončí fail-high, použije se prohledání do plné hloubky
- - redukované prohledání se aktivuje, pouze pokud je změna hloubky podstatná, tj. v průměru aspoň dané δ

Nezařazené

- Bradley-Terry model: kvalita tahu, tým fíčur
- ad PNS: rozdělení s ”heavy tail” v SAT: použi restart
- AlfaBeta a Exploitation vs. Exploration: různé nesystematické přístupy; není Anytime