Alpha-Beta-Verbesserungen

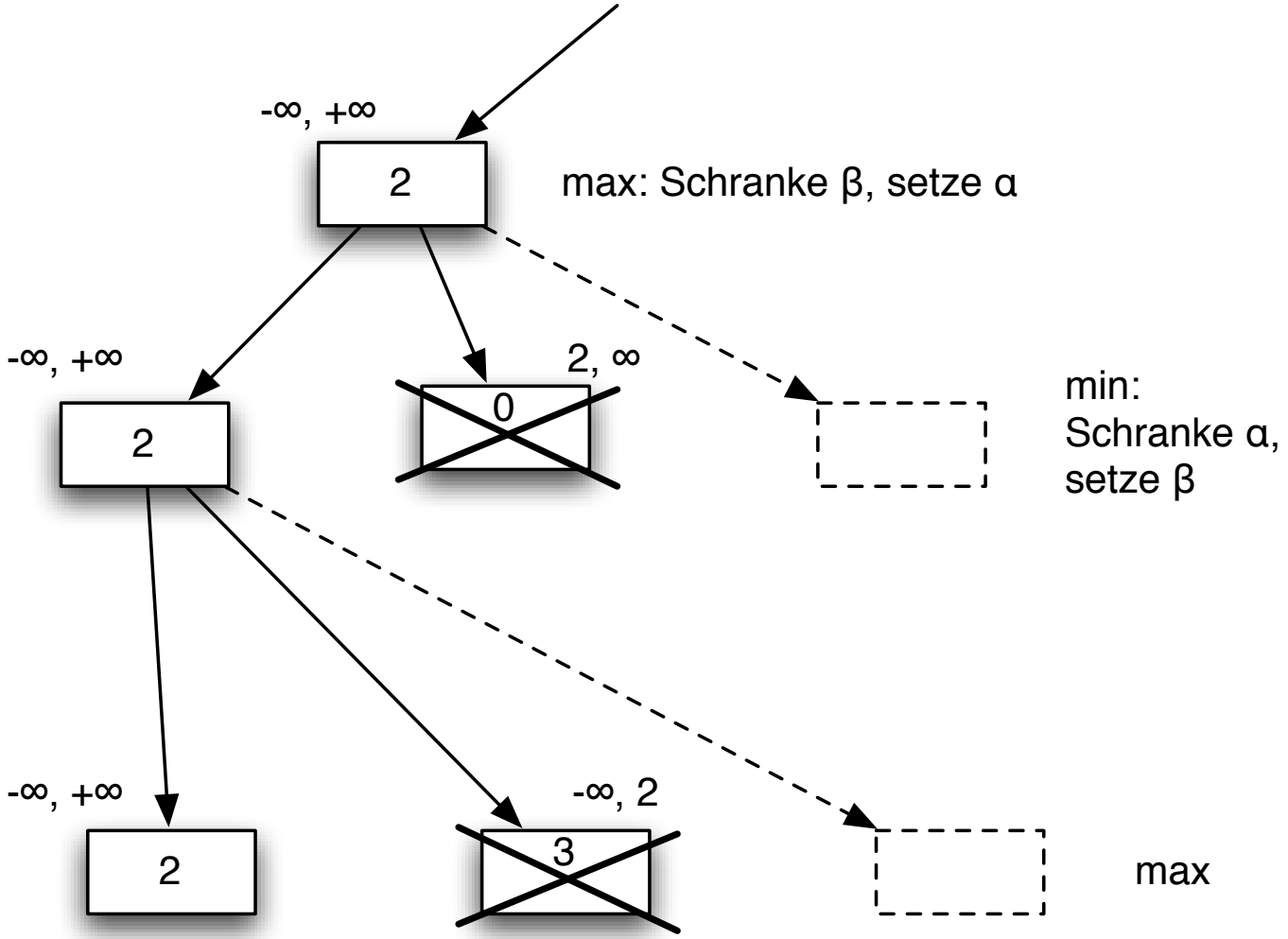
Plenum Programmierpraktikum

2007-01-18

Basis: Alpha-Beta-Pruning

- Intervall $[\alpha, \beta]$ von relevanten Werten.
- Maximierungsknoten: Setze α, Cut-Off bei Überschreitung von β.
- Minimierungsknoten: Setze β, Cut-Off bei Überschreitung von α.

Alpha-Beta-Pruning: Beispiel



Verbesserung: Negamax

- Idee: Negiere Ergebnisse der Kinder, dann kann man immer maximieren.
- Voraussetzung: Bewertung(Brett, Weiss) = – Bewertung(Brett, Schwarz).

Negamax

```
function negamax(node, depth, α, β)
    if node is a terminal node or depth = 0
        return the heuristic value of node
    else
        foreach child of node
            α := max(α, -negamax(child, depth-1, -β, -α))
            if α ≥ β      // alpha-beta pruning
                return β
        return α
```

NegaScout

- Auch: PVS (Principal Variation Search).
- Idee: Wenn die Züge gut geordnet sind, kann man sich nach dem ersten Zug Berechnungen sparen, wenn man nur versucht zu beweisen, dass die Nachfolger schlechter sind.

NegaScout

```
function negascout(node, depth, α, β)
    if node is a terminal node or depth = 0
        return the heuristic value of node
    b := β
    foreach child of node
        v := -negascout (child, depth-1, -b, -α)
        if α < v < β and not the first child
           // Nullfenster nach oben durchbrochen: re-search
           v := -negascout(child, depth-1, -β, -v)
        α := max(α, v)
        if α ≥ β
            return α // Cut-Off
        b := α+1 // Setze Nullfenster
    return α
```

Tabellen

- Generelle Idee: Wiederverwendung der Erkenntnisse vergangener Iterationen.
- Mit Tabellen ist eine (d)-Suche gefolgt von einer (d+1)-Suche meist genauso schnell wie eine (d+1)-Suche alleine!

Transposition-Table

- Im Schach kann man sich im Kreis bewegen, bei Reversi kann es (auf der selben Ebene) immer noch ähnliche Unterbäume geben.

- Man merkt sich also gemachte Bewertungen in einer Tabelle fester Größe.
  - Schlüssel: Brett kann als 2 Longs komplett kodiert werden. Verbinde diese per Addition zu einem Long, nimm diesen modulo Tabellengröße (Primzahl!).
  - Verdrängung: „tiefere“ (bessere) Werte verdrängen existierende Einträge.

- Bei der Suche: Verwende Tabellendaten zum Ordnen der Züge, speichere danach neues Ergebnis in der Tabelle.

Transposition-Table: Typischer Eintrag

- `Lock`: Stelle präzise fest, ob der Eintrag für das momentane Spielfeld passt. Beispielsweise können zwei Longs das Spielfeld exakt abspeichern.

- `Move`: Welcher Zug wurde in der aktuellen Situation gewählt?

- `Merit`: Wert der Unterbaums.

- `Flag`: Ist `Merit` eine obere Grenze, eine untere Grenze oder ein exakter Wert?
  - `VALID` (exakter Wert): Wenn die Evaluierungsfunktion aufgerufen wurde.
  - `UBOUND` (obere Grenze): Wenn β überschritten wurde (Cut-off).
  - `LBOUND` (untere Grenze): Wenn ein neues α gesetzt wurde.

- `Height`: Aus welcher Tiefe stammt `Merit`? Man muss entweder absolute Tiefen verwenden oder die Tiefen nach jedem Durchlauf anpassen.

Refutation-Table

- Transpositionstabelle benötigt viel Platz und sorgt bei großen Suchbäumen nur noch für wenig Cut-offs. Ergänze durch Refutation-Table.

- Speichere für alle Züge an der Wurzel die gewählten Pfade (die $Refutation\ Lines$), also nur noch das Feld `Move` aus der TT.

- Beginne die Suche mit diesen Pfaden, die Wahrscheinlichkeit ist damit hoch, dass dadurch (erneut) sehr viele Züge „widerlegt“ (refuted), also abgeschnitten, werden.

History-Heuristic

- Idee (vor allem für Schach relevant): Wenn ein Zug einmal gut war, dann ist er es in anderen Situationen vielleicht auch.
- Inkrementiere jedes Mal, wenn ein Zug „gut“ ist, einen Zähler. Bevorzuge Züge mit hohem Zähler beim Zugordnen.
- Hilft beim Zugordnen, wo andere Tabellen keine Ergebnisse liefern.
- Vereinfachte Version: Killer-Heuristic, merkt sich gute Züge $pro\ Ebene$.

Sonstige Ideen

- Eröffnungsbibliothek: Ab Mitte des Spiels werden Computer langsam unschlagbar.
- Lookup-Tabellen (ternär) für die Bewertungsfunktion: Kanten, 3 x 3 Ecken, 2 x 5 Ecken.
- Spielfeld als 2 longs (binär) speichern.

Sonstiges

- LobbyActions: Singleton-Pattern bei `ACTION_QUIT` und `ACTION_WAIT`.

XP Revisited

- Pair-Programming-Simulation, wenn man alleine ist: in bestimmten zeitlichen Abständen öfters über den Code gehen.

- Refactoring (keine neuen Features, Hilfe dabei: Unit-Tests) vs. Implementieren neuer Features.

- "Make it work" versus "Make it right" (wenn man mal eine erste Lösung fertig hat und komplett überblicken kann).

- Inkrementelles Entwickeln, erst bei Bedarf: You are not going to need it (YAGNI).

- Nicht zu früh abstrahieren. Das menschliche Abstraktionsverhalten ist schlechter als sein Ruf. Man kann als Mensch viel besser mit Beispielen als mit Abstraktionen umgehen.

Menschen können mit Negation nur schlecht umgehen

(Genauso wie mit Abstraktion.)

Als Mensch sieht man automatisch das Negierte vor seinem inneren Auge. Deshalb funktioniert Kritik oft nicht so gut und deshalb sollte man statt „erzähle das niemandem“ besser sagen „behalte es für dich“.

Papers (1/2)

- *A Review of Game-Tree Pruning*, T. A. Marsland, 1986.
  http://games.cs.ualberta.ca/~tony/OldPapers/1986review.pdf
  Einführung in Suchbäume.

- *The History Heuristic and Alpha-Beta Search Enhancements in Practice*, Jonathan Schaeffer, 1989.
  http://www.cs.ualberta.ca/~jonathan/Papers/ai.1989.html
  Kompakter und sehr verständlicher Überblick am Anfang.

- *The Games Computers (and People) Play*, Jonathan Schaeffer, 2000.
  http://www.cs.ualberta.ca/~jonathan/Papers/ai.2000.html
  Der Abschnitt über Reversi ist nett zu lesen.

Papers (2/2)

- *Computer Chess and Search*, T. A. Marsland, 1991.
  http://www.cs.ualberta.ca/~tony/RecentPapers/report.mac.pdf
  Beschreibt auf Seite 13 und 14 wie PVS (=NegaScout) um eine Transposition-Table erweitert werden kann, indem man die Nullfenstersucher von einer externen Funktion erledigen lässt.

- Sonstiges Material (verweist auf weitere interessante Seiten): http://www.geocities.com/h_beasley/about.html.

Aufgaben

Turnier: Wer bei der Latenz schummelt oder DoS-Attacken fährt, riskiert seinen Schein.

- NegaScout (freiwillig: Transpositionstabelle). Idee: Alte und neue Implementierung vergleichen, Ergebnis ins Forum stellen. Deadline: Endabnahme.

- Screenshot bzw. -cast der GUI. Deadline: Montag, 22. Januar.