b* probability based search

seminar, knowledge engineering und lernen in spielen, summer term 2010

lavong soysavanh

tu darmstadt
ke group
prof. j. fuernkranz

2010-05-25

TECHNISCHE UNIVERSITÄT DARMSTADT

outline

history

Figure: hans berliner, carl ebeling

recap: minimax

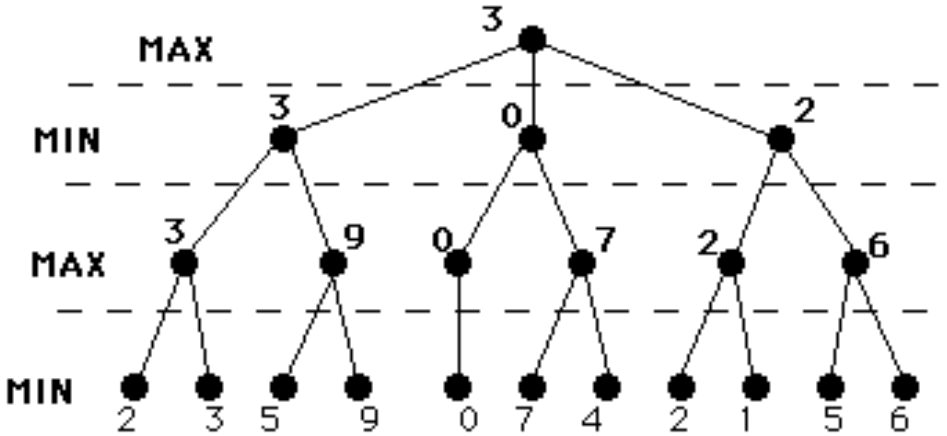

Figure: minimax algorithm

recap: alpha-beta search

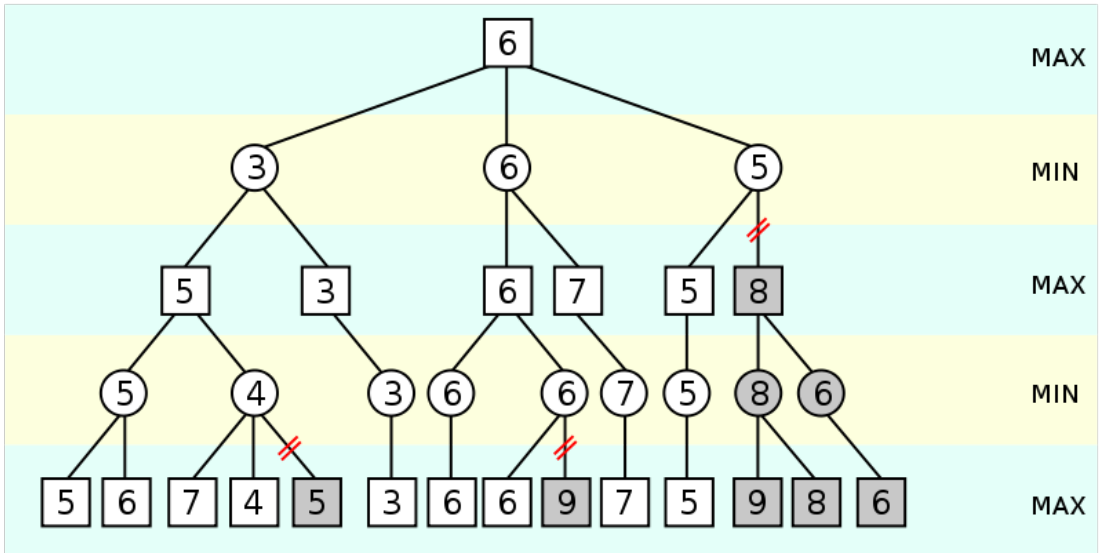

Figure: alpha-beta pruning

history

- alpha-beta search
  - undeniably efficient
  - why something else?
- selective search
  - *stop* when a clearly best alternative exists at root
  - *focus* the search on the place where the greatest information can be gained towards terminating the search
  - examples include:
    - b*

original b*

- idea
  - increase lower bound of best node (proveBest strategy)
    or
  - reduce upper bound of competing alternatives
    (disproveRest strategy)
    until *separation* occurs
- problems with the original b*
  - static evaluation function - *threat* detection?
  - distribution of likely values within range need not be uniform

range not uniform

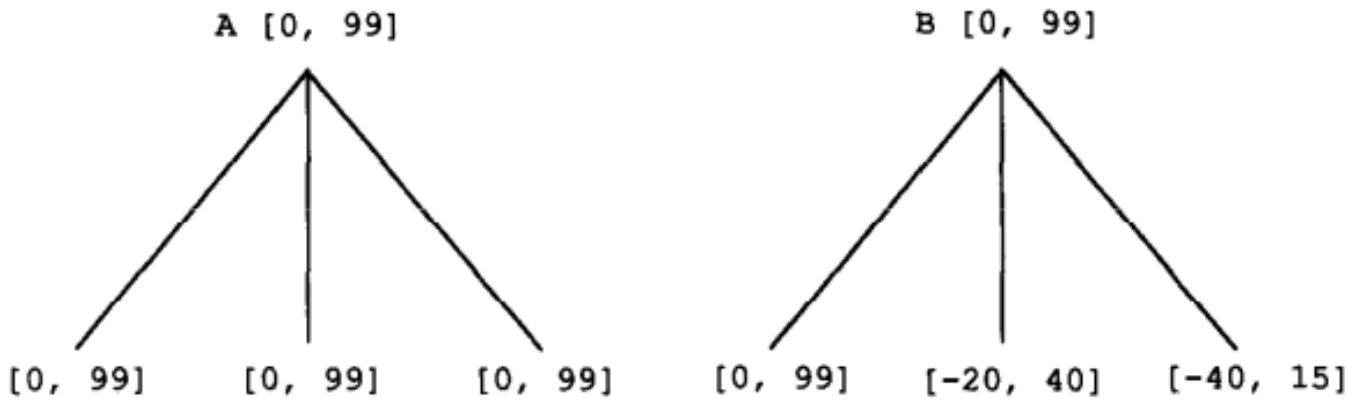

Fig. 3. Bounds are not sufficient to portray goodness.

improvements made

- use shallow searches to produce quite accurate bounding values
- use extra move for the player at the start of search to get optimistic value
- introduce notation of *realistic value* and partition the range of a node into two segments
  - range between optimistic and realistic value (*domain* of player)
  - range between realistic and pessimistic value (*domain* of opponent)
- use a probability distribution instead of a range as the value of a node, and produce a calculus for backing up distributions

representation of a b* node

- *RealVal*, best estimate of the true value of a node (determines which move is considered better)
- *OptVal*, optimistic value of a node for the side-on-move (drives the search)
- *PessVal*, optimistic value of a node for the side-not-on-move
- *OptPrb*, probability the a certain value can be achieved by future searches of the subtree rooted at this node

two step search

- *select*
  - is to identify best move for player
  - examines players *optimism* (*OptPrbs* define players potential at each node)
  - terminate early if find a node to be 'clearly best' (OptPrb valued in such a way that it's unlike to achieve as good a RealVal)
- *verify*
  - is to show that selected move is not best
  - find opponents reply that is good enough to reduce RealVal of selected move so that it's no longer best
  - terminate early if none remaining moves to be investigated appear to have potential for refutation
- both phases governed by an effort limit

node selection

- roles
  - *forcer* - tries to move that has the greatest chance of achieving some optimistic value
  - *obstructor* - tries to limit the effective of forcer's move
  - *targetVal* - level of success that forcer tries to achieve
  - role exchange: in select phase player is forcer and in verify phase opponent is forcer
- targetVal
  - in *select* phase - always greater than best RealVal, adjusted as RealVal changes
  - in *verify* phase - remains at the value that the RealVal of selected move is to reduced to

effort limits

- types of *effort limit*
  - amount of time for investigation
  - closeness of competing alternatives
  - likelihood of achieving TargetVal
- set effort limit at start based on size of the tree and the average amount of time until available per move

probability based b* search

```
integer TargetVal;
SELECT: while (RealVal(BestMoveAtRoot)<
               OptVal(AnyOtherMoveAtRoot))
        {
               TargetVal=(OptVal(2ndBest)+RealVal(Best))/2;
               Select Root Node with greatest OptPrb;
               Trace down the child subtree selecting
                      For Player-to-Move nodes, child with
                      largest OptPrb
                      For Opponent-to-Move nodes, child with
                      best RealVal
         Until arriving at a leaf node;
         Get RealVal for each Child Node of this leaf;
         If it is a Player-to-Move node get OptVals for each
         Child;
         Back up Values;
         if (EffortLimitsExceeded) Break;
        }
```

probability based b* search (cont.)

```
TargetVal=RealVal(2ndBestMoveAtRoot)-1;
VERIFY: while (RealVal(BestMoveAtRoot)>=TargetVal)
        {
         Select Reply Node with greatest OptPrb;
         Trace down the child subtree selecting
                For Opponent-to-Move nodes, child with largest
                OptPrb
                For Player-to-Move nodes, child with best Real-
                Val
         Until arriving at a leaf node;
         Get RealVal for each Child Node of this leaf;
         If it is an Opponent-to-Move node get OptVals for each
         Child;
         Back up Values;
         if (EffortLimitsExceeded) GoTo MakeMove; !! It passed
         inspection.
        }
GoTo SELECT;                    !! Selected move was refuted.
MakeMove:
```

rationale for a working b* search

- comparisons among alternatives with a view toward terminating when a clearly best alternative emerges
- use of *optimism* to guide the search
- finding lines of play based upon simple resistance by the opponent before allowing him to venture to find a refutation
- using probability distribution to capture the notion of goodness

parallelization

(1) B* performance has already been shown to increase with additional nodes investigated, and depth of probe searches over a modest range.
(2) B* decomposes easily for parallelism.
(3) Potential losses in efficiency come from having to predict which nodes need to be expanded, and there may be some loss if that node is expanded but would never have been if a single machine were at work. We estimate the magnitude of such losses.
(4) We review the literature on parallel decomposition of alpha-beta search and what the efficiency losses are.
(5) Since the losses due to alpha-beta are on the order of 90% and the losses due to B* are on the order of 15%, we conclude that as computing power increases, B* searches must overtake alpha-beta searches.

no horizon effect on b*

- *horizon effect*
  - misleading result due to search with limited depth for feasibility reasons - making a move after five ply search may prove to be detrimental after the 6th
- i-have-optimism-that-needs-to-be-investigated attitude will notice it when getting into *verify* phase and the effect is reasonably probable
- "thus horizon effects, which push undesirable results over the search horizon, do not seem to occur in b* searches"

b* hitech tournament results

Table 1
B* Hitech tournament results

Tourney	Result	Place
7th World Computer Champ. 1992	3 – 2	7th out of 22.
ACM Internat. Computer Champ. 1993	3 – 2	3rd out of 12.
AEGON Human-Computer Tourn. 1993	3 – 3[a]	14th out of 32.

[a] Some of these games were played by Hitech 5.6 as bugs were found and corrected in B* Hitech.

for further reading


hans j. berliner, chris mcconnell
b probability based search*.
http://dx.doi.org/10.1016/0004-3702%2895%2900092-5


chessprogramming wiki
https://chessprogramming.wikispaces.com/B*