Ss State-space Search

Adversarial Search III:
Increasing Game-Playing Performance --
Zobrist Hashing, and Machine Learning in Checkers

CSE 415: Introduction to Artificial Intelligence
University of Washington
Winter, 2019

Ss State-space Search

Outline

- AI and Checkers in the Early Days
- Additional game-playing issues
- Learning a scoring polynomial from experience (example from Checkers)
- Zobrist Hashing
- AI and Games Today

Ss State-space Search

The Thinking Machine (1960's)

Ss State-space Search

Game-Playing Issues

Representing moves: a (Source, Destination) approach works for some games when the squares on the board have been numbered.
Source: The number of the square where a piece is being moved from.
Destination: The number of the square where the piece is being moved to.
(For Othello, only the destination is needed.)

Opening moves:
Some programs use an "opening book"
Some competitions require that the first 3 moves be randomly selected from a set of OK opening moves, to make sure that players are "ready for anything"

Regular maximum ply are typically 15-20 for machines, with extra ply allowed in certain situations.

ply = # of moves to look ahead; no rigid cutoff usually but should rely on given situation to determine how far to go (i.e. during a long trade sequence in chess). Keep going until you reach some steady state.

Static evaluation functions in checkers or chess may take 15 to 20 different features into consideration.

Use iterative deepening; don't want to run out of time; want to explore as deeply as possible

Ss State-space Search

Learning a Scoring Polynomial From Experience

Arthur Samuel: Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, Vol 3. pp.211-229, 1959.
http://www.research.ibm.com/journal/rd/033/ibmrd0303B.pdf

Arthur Samuel: Some Studies in Machine Learning Using the Game of Checkers. II --- Recent Progress. *IBM Journal*, Vol 116. pp.601-617, 1967.
http://www.research.ibm.com/journal/rd/116/ibmrd1106C.pdf

Ss State-space Search

Scoring Polynomial

$$f(s) = a_1 \text{ ADV} + a_2 \text{ APEX} + a_3 \text{ BACK} + \ldots + a_{16} \text{ THRET}$$

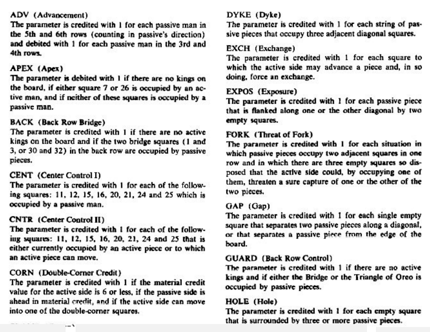
Had computer learn how to set the static evaluation function and tweak coefficients and terms based on feedback from learning games. Rather than setting some arbitrary function defined by us.

There are 16 terms at any one time.
They are automatically selected from a set of 38 candidate terms.

26 of them are described in the following 3 slides.

Ss
State-space
Search

Scoring Polynomial Terms



ADV (Advancement)
The parameter is credited with 1 for each passive man in the 5th and 6th rows (counting in passive's direction) and debited with 1 for each passive man in the 3rd and 4th rows.

APEX (Apex)
The parameter is debited with 1 if there are no kings on the board, if either square 7 or 26 is occupied by an active man, and if neither of these squares is occupied by a passive man.

BACK (Back Row Bridge)
The parameter is credited with 1 if there are no active kings on the board and if the two bridge squares (1 and 3, or 30 and 32) in the back row are occupied by passive pieces.

CENT (Center Control I)
The parameter is credited with 1 for each of the following squares: 11, 12, 15, 16, 20, 21, 24 and 25 which is occupied by a passive man.

CNTR (Center Control II)
The parameter is credited with 1 for each of the following squares: 11, 12, 15, 16, 20, 21, 24 and 25 that is either currently occupied by an active piece or to which an active piece can move.

CORN (Double-Corner Credit)
The parameter is credited with 1 if the material credit value for the active side is 6 or less, if the passive side is ahead in material credit, and if the active side can move into one of the double-corner squares.

DYKE (Dyke)
The parameter is credited with 1 for each string of passive pieces that occupy three adjacent diagonal squares.

EXCH (Exchange)
The parameter is credited with 1 for each square to which the active side may advance a piece and, in so doing, force an exchange.

EXPOS (Exposure)
The parameter is credited with 1 for each passive piece that is flanked along one or the other diagonal by two empty squares.

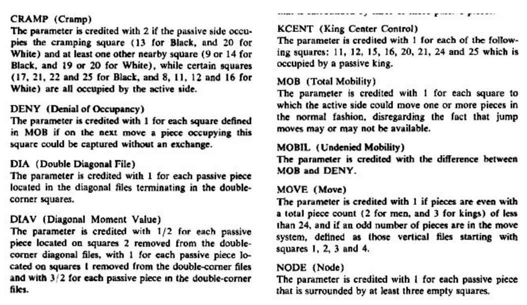
FORK (Threat of Fork)
The parameter is credited with 1 for each situation in which passive pieces occupy two adjacent squares in one row and in which there are three empty squares so disposed that the active side could, by occupying one of them, threaten a sure capture of one or the other of the two pieces.

GAP (Gap)
The parameter is credited with 1 for each single empty square that separates two passive pieces along a diagonal, or that separates a passive piece from the edge of the board.

GUARD (Back Row Control)
The parameter is credited with 1 if there are no active kings and if either the Bridge or the Triangle of Oreo is occupied by passive pieces.

HOLE (Hole)
The parameter is credited with 1 for each empty square that is surrounded by three or more passive pieces.

Ss
State-space
Search



CRAMP (Cramp)
The parameter is credited with 2 if the passive side occupies the cramping square (13 for Black, and 20 for White) and at least one other nearby square (9 or 14 for Black, and 19 or 20 for White), while certain squares (17, 21, 22 and 25 for Black, and 8, 11, 12 and 16 for White) are all occupied by the active side.

DENY (Denial of Occupancy)
The parameter is credited with 1 for each square defined in MOB if on the next move a piece occupying this square could be captured without an exchange.

DIA (Double Diagonal File)
The parameter is credited with 1 for each passive piece located in the diagonal files terminating in the double-corner squares.

DIAV (Diagonal Moment Value)
The parameter is credited with 1/2 for each passive piece located on squares 2 removed from the double-corner diagonal files, with 1 for each passive piece located on squares 1 removed from the double-corner files and with 3/2 for each passive piece in the double-corner files.

KCENT (King Center Control)
The parameter is credited with 1 for each of the following squares: 11, 12, 15, 16, 20, 21, 24 and 25 which is occupied by a passive king.

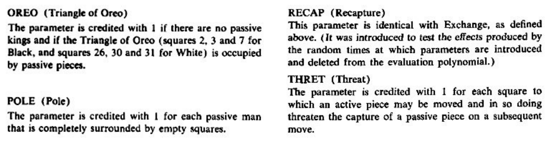
MOB (Total Mobility)
The parameter is credited with 1 for each square to which the active side could move one or more pieces in the normal fashion, disregarding the fact that jump moves may or may not be available.

MOBIL (Undenied Mobility)
The parameter is credited with the difference between MOB and DENY.

MOVE (Move)
The parameter is credited with 1 if pieces are even with a total piece count (2 for men, and 3 for kings) of less than 24, and if an odd number of pieces are in the move system, defined as those vertical files starting with squares 1, 2, 3 and 4.

NODE (Node)
The parameter is credited with 1 for each passive piece that is surrounded by at least three empty squares.

Ss
State-space
Search



OREO (Triangle of Oreo)
The parameter is credited with 1 if there are no passive kings and if the Triangle of Oreo (squares 2, 3 and 7 for Black, and squares 26, 30 and 31 for White) is occupied by passive pieces.

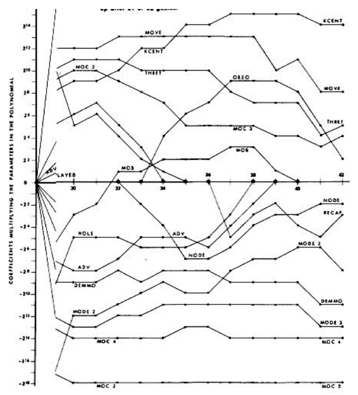
POLE (Pole)
The parameter is credited with 1 for each passive man that is completely surrounded by empty squares.

RECAP (Recapture)
This parameter is identical with Exchange, as defined above. (It was introduced to test the effects produced by the random times at which parameters are introduced and deleted from the evaluation polynomial.)

THRET (Threat)
The parameter is credited with 1 for each square to which an active piece may be moved and in so doing threaten the capture of a passive piece on a subsequent move.

Ss
State-space
Search

Scoring Polynomial Coefficient Adjustment



Coefficients are powers of 2.

They are ordered so that no two are equal at any time.

Ss
State-space
Search

Polynomial adjustment

For each term, the program keeps track of whether its value was correlated with an improvement in the game position over a series of moves.

keep track of how well each variable judges the system game after game and adjust its weighting accordingly

If so, its value goes up, if not, it goes down.

Ss
State-space
Search

Checkers: Computer vs Human

Samuel's program beat a human player in a widely publicized match in 1962.

Later a program called Chinook, developed by Jonathan Schaeffer at the Univ. of Alberta became the nominal "Man vs Machine Champion of the World" in 1994.

Checkers playing was the vehicle under which much of the basic research in game playing was developed.

Zobrist Hashing: Another Performance Technique

iterative deepening has a lot of repeated computation; using hashing to keep track of already computed states to avoid extra computations

Avoid recomputing values for some states (especially those within 3 or 4 ply of the current state, which are relatively expensive to compute), by saving their values.

Use a hash table to save: [state, value, ply-used].
As a hashing function, use a *Zobrist hashing function*:

the bit operation

For each piece on the board, exclusive-or the current key with a pre-generated random number.

for checkers, check the state of each square on the board. (empty, white, black, white king, black king)

Hash values for similar boards are very different.
Hash values can be efficiently computed with an incremental approach (in some games, like checkers and chess, at least).

idea. When the state changes, all you have to do is exclusive or the parts of the board that changes. WONT have to reinitialize entire board.

Zobrist Hashing in Python

```
# Set up a 64x2 array of
# random ints.
S = 64
P = 2
zobristnum =\
 [[0]*P for i in range(S)]
from random import randint

def myinit():
  global zobristnum
  for i in range(S):
    for j in range(P):
      zobristnum[i][j]=\
        randint(0, \
         4294967296)

myinit()
```

```
# Hash the board to an int.
def zhash(board):
  global zobristnum
  val = 0;
  for i in range(S):
    piece = None
    if(board[i] == 'B'): piece = 0
    if(board[i] == 'W'): piece = 1
    if(piece != None):
      val ^= zobristnum[i][piece]
    return val

# Testing:
b = [' ']*64 ; b[0]='B' ; b[1]='W'

print(zhash(b))
```

bit operation

```
3473306553
```

Artificial Intelligence in Games Today

AI techniques get incorporated into games in many ways, from automatic creation of environments and characters to modeling of the human players.

This has become a subfield of its own, at the intersection of AI and Game Technologies.

Artificial Intelligence and Games
A Springer Textbook | By Georgios N. Yannakakis and Julian Togelius, 2018.