Home * Evaluation * Game Phases * Endgame * Pawn Endgame * **Blockage Detection**

4k3/8/3pPp2/1p1P1P1p/1P5P/5P2
/3K4/8 w - -

Blockage Detection, a pattern recognition technique to statically discover rammed positions in pawn endgames, and permanently prevented kings from penetrating into opponent's camp. In those positions material advantage is not sufficient to win such games, otherwise only recognizably by a very deep search, when a draw by fifty-move rule rises the horizon. Best not only applied in leaf node evaluation, but as interior node recognizer within the search, a detected blockade either prunes forward or narrows the search window.

Table of Contents

Search Application

Ram Fence

Blockage with Dynamic Pawns

See also

Publications

Forum Posts

References

What links here?

Search Application

Blockage detection is only applied for the side with advantage, proving its upper bound is a draw score [1]:

```
int search(int alpha, int beta, ...) {
  ...
  /* blockade detection */
  if ( beta > DRAW_SCORE ) {
    if ( blockage() ) {
      if ( alpha >= DRAW_SCORE )
        return DRAW_SCORE;
      beta = DRAW_SCORE;
    }
  }
  /* continue regular search */
}
```

Ram Fence

Following sample assumes White the "winning" side to move

The blockage requires a fence consisting of own fixed pawns and opponent pawn attacks, to divide the board into two disconnected regions. There are various algorithms to deal with fence detection, and fixed and dynamic pawns, following fill based proposal deals with bitboards and should only take none lever pawns into account to prove the fence is impenetrable. First cheap condition is the presence of at least three rams. Further own pawns blocked by own rams are included successively into the fence set, and finally opponent pawn attacks. The precondition, the defending king has at least one vacant square to avoid zugzwang issues, is omitted.

A king flood starts at its base rank (1st rank for White) over the board (ignoring black passers on the 2rd rank), or all bits below the fence' least significant one bit, the potential fence as flood stopping obstruction. If, after a few cycles, the flood reaches the above area, the fence is penetrable, and the blockage test failed. Each time, the king flood drowns (undefended) opponent pawns, the ram-fence is re-calculated with the drown pawns "captured". If the flood does not grow anymore, the fence is detected.

see One Step, Population Count *and* BitScan

```
bool forms_fence4white(
U64 wPawns, U64 bPawns, U64 &fence, U64 &flood) {
  fence = wPawns & soutOne( bPawns );
  if (popCount( fence ) < 3)
    return false;
  fence |= wPawns & soutOne( fence );
  fence |= wPawns & soutOne( fence );
  fence |= wPawns & soutOne( fence );
  fence |= wPawns & soutOne( fence );
  fence |= bPawnAttacks( bPawns );
  flood = RANK1BB | allBitsBelow[BitScanForward( fence )];
```

```
  U64 above = allBitsAbove[BitScanReverse( fence )];
  while ( true ) {
    U64 temp = flood;
    flood |= eastOne( flood ) | westOne( flood );
/* Set all 8 neighbors */
    flood |= soutOne( flood ) | nortOne( flood );
    flood &= ~fence;
    if (flood == temp)
      break; /* Fill has stopped, blockage? */
    if (flood & above) /* break through? */
      return false; /* yes, no blockage */
    if (flood & bPawns) {
      bPawns &= ~flood;  /* "capture" undefended black pawns */
      fence = wPawns & soutOne( bPawns );
      if (popCount( fence ) < 3)
        return false;
      fence |= wPawns & soutOne( fence );
      fence |= wPawns & soutOne( fence );
      fence |= wPawns & soutOne( fence );
      fence |= wPawns & soutOne( fence );
      fence |= bPawnAttacks( bPawns );
    }
  }
  return true;
}
```

In order to ensure that White cannot penetrate the fence, the white kings must reside in the below flood set, and the defending, black king outside the below set. If all pawns of the "winning" side are rammed without any lever, the most obvious case of a blockage is recognized.

```
. . . k . . . .          a a a a a a a a
. . . . . . . .  above   a a a a a a a a
. . . . . . . .          a a a a a a a a
. b . . b . . b          a a a a a a a a
. w . . w . . w  fence    F_F_F_F_F_F_F_F
. w . . . . . .          b f b b b b b b
. . . K . . . .  below    b b b b b b b b
. . . . . . . .  flood    b b b b b b b b
```

Blockage with Dynamic Pawns

Otherwise, one has to consider dynamic pawns of the "winning" side, able to push forward or to capture, in

particular above the fence, as member of the fence (levers) and below the fence (from White's point of view). For instance, one (protected) passed pawn, with the defending king on its front span, is not sufficient to break the blockage. Similar holds for pawns below the fence not able to lever, and becoming rammed if moving forward, or with some care, most backward pawns.

```
. . . . k . . .          a a a a a a a a
. . . . . . . .  above   a a a a a a a a
. . . b w b . .          a a a a a a a a
. b . w . w . b          a a F_F_F_F_F a
. w . . . . . w  fence   F_F_F b b b F_F
. . . . . w . .          b b b b b b b b
. . . K . . . .  below   b b b b b b b b
. . . . . . . .  flood   b b b b b b b b
```

More complicated cases for a successful blockage detection also with levers involved are elaborated in Omid David's et al. 2004 ICGA Journal paper [2], also published as code of Chiron by Ubaldo Andrea Farina in CCC [3].

See also

- Corresponding Squares
- Draw Evaluation
- Flood Fill Algorithm
- Fortress
- Interior Node Recognizer
- Pattern Recognition
- Chunking

Publications

- Peter W. Frey, Larry Atkin (**1979**). Creating a Chess-Player, Part 4: Thoughts on Strategy. In Blaise W. Liffick (ed.), The Byte Book of Pascal, pp. 143-155. Byte Publications, also BYTE, Vol. 4, No. 1
- Omid David, Ariel Felner, Nathan S. Netanyahu (**2004**). Blockage Detection in Pawn Endings. CG 2004, pdf
- Omid David, Ariel Felner, Nathan S. Netanyahu (**2004**). *Blockage Detection in Pawn Endgames.* ICGA Journal, Vol. 27, No. 3

Forum Posts

- Endgame Blockage Positions by Omid David, CCC, December 06, 2002

- Jeremiah's Wall Detection by Renze Steenhuisen, CCC, April 28, 2005
- Code of Chiron for Detection of Pawn Blockages by Ubaldo Andrea Farina, CCC, October 13, 2011
- To users of Chiron by Richard Vida, CCC, January 02, 2012
- Details about Critter 1.4a. by Jesús Muñoz, CCC, February 10, 2012 » Critter

References

1. ^ Code snippet from Omid David, Ariel Felner, Nathan S. Netanyahu (**2004**). Blockage Detection in Pawn Endings. CG 2004, pdf
2. ^ Omid David, Ariel Felner, Nathan S. Netanyahu (**2004**). *Blockage Detection in Pawn Endgames*. ICGA Journal, Vol. 27, No. 3
3. ^ Code of Chiron for Detection of Pawn Blockages by Ubaldo Andrea Farina, CCC, October 13, 2011

What links here?

Page	Date Edited
BlackMamba	Nov 26, 2016
Blockade	Oct 19, 2017
Blockade of Stop	Oct 11, 2016
Blockage Detection	Oct 19, 2017
Chiron	Sep 24, 2017
Critter	Jan 25, 2014
Draw	Apr 14, 2018
Draw Evaluation	Mar 12, 2018
Endgame	Sep 18, 2017
Falcon	Sep 2, 2016
Fortress	Feb 1, 2018
Hossa	Jan 7, 2016
Interior Node Recognizer	Mar 12, 2018
Jan Renze Steenhuisen	Sep 10, 2017
Mate Search	Oct 22, 2016
Pawn Endgame	Oct 11, 2017
Pawn Rams (Bitboards)	Nov 11, 2017
Pawn Structure	Oct 2, 2017
Vajolet	Jan 25, 2018

Up one level