The way it used to be:

bool MPC(Position position, int alpha, int depth, int selectivity, int & value)

{

int bound;

if (selectivity == 0) return false;

for (int i = 0; i < NUM\_TRY; i++)

{

if (depth == Depth[i])

{

bound = round(alpha + T[selectivity] \* sigma[i]);

if (ZWS(position, bound-1, depth[i], 0) >= bound){

value = alpha+1;

return true;

}

bound = round(alpha - T[selectivity] \* sigma[i]);

if (ZWS(position, bound, depth[i], 0) <= bound){

value = alpha;

return true;

}

}

}

return false;

}

int ZWS(Position position, int alpha, int depth, int selectivity)

{

int value;

if (depth == 0) return Evaluate(position);

if (TTCut(position, value))

return value;

if (MPC(position, alpha, depth, selectivity, value))

return value;

for\_each(move in Movelist)

{

if (-ZWS(position.play(move), -alpha-1, depth-1, selectivity) > alpha)

{

UpdateTT(position, depth, selectivity, alpha+1, SCORE\_MAX);

return alpha+1;

}

}

UpdateTT(position, depth, selectivity, SCORE\_MIN, alpha);

return alpha;

}

int PVS(Position position, int alpha, int beta, int depth, int selectivity)

{

int value;

bool SearchPV = true;

if (depth == 0) return Evaluate(position);

if (TTCutOff(position, value))

return value;

for\_each(move in Movelist)

{

if (SearchPV)

value = -PVS(position.play(move), -beta, -alpha, depth-1, selectivity);

else

{

value = -ZWS(position.play(move), -alpha-1, depth-1, selectivity);

if (value > alpha)

value = -PVS(position.play(move), -beta, -alpha, depth-1, selectivity);

}

if (value >= beta)

{

UpdateTT(position, depth, selectivity, beta, SCORE\_MAX);

return beta;

}

if (value > alpha)

{

alpha = value;

SearchPV = false;

}

}

if (SearchPV)

UpdateTT(position, depth, selectivity, SCORE\_MIN, alpha);

else

UpdateTT(position, depth, selectivity, alpha, alpha);

return alpha;

}

My improvements:

int ZWS(Position position, int alpha, int depth, int selectivity, bool & GotProbCut)

{

int value;

bool GetsProbCut;

if (depth == 0) return Evaluate(position);

if (TTCut(position, value, GotProbCut))

return value;

if (GotProbCut = MPC(position, alpha, depth, selectivity, value))

return value;

for\_each(move in Movelist)

{

GetsProbCut = false;

value = -ZWS(position.play(move), -alpha-1, depth-1, selectivity, GetsProbCut);

GotProbCut |= GetsProbCut;

if (value > alpha)

{

if (GetsProbCut)

UpdateTT(position, depth, selectivity, alpha+1, SCORE\_MAX);

else

UpdateTT(position, depth, NO\_SELECTIVITY, alpha+1, SCORE\_MAX);

return alpha+1;

}

}

if (GotProbCut)

UpdateTT(position, depth, selectivity, SCORE\_MIN, alpha);

else

UpdateTT(position, depth, NO\_SELECTIVITY, SCORE\_MIN, alpha);

return alpha;

}

int PVS(Position position, int alpha, int beta, int depth, int selectivity, bool & GotProbCut)

{

int value;

bool GetsProbCut;

bool SearchPV = true;

if (depth == 0) return Evaluate(position);

if (TTCutOff(position, value))

return value;

for\_each(move in Movelist)

{

if (SearchPV)

{

GetsProbCut = false;

value = -PVS(position.play(move), -beta, -alpha, depth-1, selectivity, GetsProbCut);

GotProbCut |= GetsProbCut;

}

else

{

GetsProbCut = false;

value = -ZWS(position.play(move), -alpha-1, depth-1, selectivity, GetsProbCut);

GotProbCut |= GetsProbCut;

if (value > alpha)

{

GetsProbCut = false;

value = -PVS(position.play(move), -beta, -alpha, depth-1, selectivity, GetsProbCut);

GotProbCut |= GetsProbCut;

}

}

if (value >= beta)

{

if (GotProbCut)

UpdateTT(position, depth, selectivity, beta, SCORE\_MAX);

else

UpdateTT(position, depth, NO\_SELECTIVITY, beta, SCORE\_MAX);

return beta;

}

if (value > alpha)

{

alpha = value;

SearchPV = false;

}

}

if (SearchPV)

{

if (GotProbCut)

UpdateTT(position, depth, selectivity, SCORE\_MIN, alpha);

else

UpdateTT(position, depth, NO\_SELECTIVITY, SCORE\_MIN, alpha);

}

else

{

if (GotProbCut)

UpdateTT(position, depth, selectivity, alpha, alpha);

else

UpdateTT(position, depth, NO\_SELECTIVITY, alpha, alpha);

}

return alpha;

}