Enhanced Realization Probability Search

MARK H.M. WINANDS*

MICC-IKAT Games and AI Group, Faculty of Humanities and Sciences,
Universiteit Maastricht,
P.O. Box 616, 6200 MD Maastricht, The Netherlands
**E-mail: m.winands@micc.unimaas.nl*

YNGVI BJÖRNSSON

Department of Computer Science, Reykjavík University,
Ofanleiti 2 IS-103 Reykjavík, Iceland
E-mail: yngvi@ru.is

In this paper we show that Realization Probability Search (RPS) significantly improves the playing strength of a world-class LOA-playing program, even when used in combination with existing state-of-the-art search enhancements. Furthermore, we introduce a much improved variant of RPS, called Enhanced Realization Probability Search (ERPS). The new algorithm addresses two weaknesses of RPS, resulting in both more robust tactical play and reduced search overhead. Our experiments in the domain of LOA show that ERPS offers just as significant improvement over regular RPS, as the latter improves upon regular search.

Keywords: Search; Heuristics; Games.

1. Introduction

The alpha-beta ($\alpha\beta$) algorithm[4] is the standard search procedure for playing board games such as chess and checkers (and many others). The playing strength of programs employing the algorithm depends greatly on how deep the search is able to explore critical lines of play. Therefore, over the years, many techniques for augmenting alpha-beta search with a more selective tree expansion mechanism have been developed, so called *variable-depth search* techniques. Promising lines of play are explored more deeply (search extensions), at the cost of others less interesting that are cut off prematurely (search reductions or forward pruning).

One recent addition to the family of variable-depth search techniques is *Realization Probability Search (RPS)*, introduced by Tsuoroka[10] in 2002.

Using this method his program, GEKISASHI, won the 2002 World Computer Shogi Championship, resulting in the algorithm gaining a wide acceptance in computer Shogi. Although researchers have experimented with the algorithm in other game domains with some success,[2] the question still remains how well the RPS enhancement works in those domains when used in a state-of-the-art game-playing program in combination with other search enhancement schemes.

In this paper we investigate the use of RPS in the game of *Lines of Action (LOA)*. The contributions of this work are twofold. First, we demonstrate the usefulness of the RPS scheme in the domain of LOA by successfully applying it in a world-class program, effectively raising the level of state-of-the-art game-play in that domain. Secondly, we introduce an important enhancement to the RPS algorithm to overcome problems it has with endgame positions. This improvement, which we label *Enhanced Realization Probability Search (ERPS)*, greatly improves the effectiveness of RPS in our game domain.

2. Realization Probability Search

The Realization Probability Search (RPS)[10] algorithm is a new way of using fractional-ply extensions.[3,5] The algorithm uses a probability based approach to assign fractional-ply weights to move categories, and then uses re-searches to verify selected search results.

The probability that a move belonging to a given move category will be played is called *transition probability*, and can be estimated beforehand from game records of games played between expert players as follows:

$$P_c = \frac{n_{played(c)}}{n_{available(c)}} \tag{1}$$

where $n_{played(c)}$ is the number of game positions in which a move belonging to category c was played, and $n_{available(c)}$ is the number of positions in which moves belonging to category c were available.

Originally, the *realization probability* of a node represented the probability that the moves leading to the node will be played. By definition, the realization probability of the root node is 1. The transition probabilities are used to compute the realization probability of a node in a recursive manner (by multiplying them). If the resulting realization probability becomes smaller than a predefined threshold, the expansion of that branch is stopped. Since a probable move has a large transition probability while an improbable has a small probability, the search proceeds deeper after probable moves and shallower after improbable moves. However, instead of using

the transition probabilities that way, we transform them into fractional plies as follows as suggested by Tsuoroka *et al.*:[10]

$$FP = \frac{log(P_c)}{log(C)} \tag{2}$$

where C is a game dependent constant between 0 and 1.

One potential pitfall of this scheme is that a player may "push" unavoidable problems beyond the search horizon by deliberately playing an inferior move with a high fractional-ply value, thus curtailing the look-ahead depth prematurely. To avoid this problem, the algorithm performs a *deeper* re-search for moves whose values are larger than the current best value (i.e., the α value), using only a small reduction value, called *minFP*.

3. Enhanced Realization Probability Search

In this section we introduce Enhanced Realization Probability Search (ERPS), an improved variant that is tactically much safer than regular RPS, while still maintaining its positional benefits. We identify two problems with RPS:

(1) A move causing a fail-low will always be ignored. A tactical move belonging to a move category with a high reduction factor may be wrongly pruned away because of a horizon effect. In this case there is a risk that the best move will not be played.
(2) A move causing a fail-high will always be re-searched with minimal FP. A weak move may be unnecessarily extended because of a horizon effect. In this case valuable computing resources are wasted, resulting in a shallower nominal search depth.

Our enhancements aim at overcoming the aforementioned problems with RPS. The first one is based on an idea proposed by Björnsson *et al.*;[1] they suggest that pruning should take place only after the first m moves have been searched (i.e., looking at the rank of the move in the move ordering). This principle is applied in recent pruning techniques, such as LMR[9] and RankCut.[6] Thus, to alleviate the first of the aforementioned problems of RPS, we cap the reduce depth of the first m moves to a maximum value t. This avoids aggressive reductions for early moves.

The second enhancement improves on how re-searches are done. Instead of performing a re-search right away to a full depth, we first interleave a shallower intermediate re-search (using a depth reduction that is the average of the original reduce depth and minFP). If that re-search also results

in a fail-high, only then is the full-depth re-search performed. We restrict these intermediate re-searches to cases where there is a substantial difference between the original and full re-search depths (the averaged decrease depth is larger than a predefined threshold *delta*). This prevents insignificant explorations.

A pseudo-code of PVS/NegaScout alpha-beta variant[7,8] enhanced with ERPS is shown in Figure 1, providing the details of the implementation. For clarity we show only the main loop of the algorithm and omit other details.

```
ERPS( state, alpha, beta, depth ){
    ... details omitted ...
    while(next != null){
      alpha = max(alpha, best);
      decDepth = FP(next);
      //Enhancement 1
      if(decDepth > t && moveCounter <= m)
            decDepth = t;
      //Preliminary Search Null-Window Search Part
      value = -RPS(next, -alpha-1, -alpha, depth-decDepth);
      //Re-search
      if(value > alpha){
        //Enhancement 2
        dec_depth = (minFP + dec_depth)/2;
        if(dec_depth>delta)
          value = -RPS(next, -alpha-1, -alpha, depth-decDepth);
        if(value > alpha)
          value = -RPS(next, -beta, -alpha, depth-minFP);
      }
      if(value > best){
        best = value;
        if(best >= beta) break;
      }
      next = nextSibling(next);
    }
    ... details omitted ...
    return best;
}
```

Fig. 1. ERPS: Two enhancements for Realization Probability Search.

Table 1. Comparing the search algorithms on 286 test positions.

Algorithm	# of positions solved
Classic	186
RPS	130
ERPS	215

4. Experiments

We empirically evaluated the RPS algorithmic enhancements in the game of Lines of Action (LOA) using the state-of-the art game-playing program MIA, which won the LOA competition at the Computer Olympiad[11] several times. The default search engine, called "Classic", is based on alpha-beta search (the PVS/NegaScout variant). We augmented it with realization probability search: RPS on the one hand and ERPS on the other, resulting in two new versions (labelled RPS and ERPS below). For all experiments we used parameter setting of (t=1, m=5, and $delta$=1.5) for EPRS.

In the first set of experiments Classic, RPS and ERPS were tested on a test-suite of 286 forced-win LOA positions. For each problem the programs were allowed to search maximum 50 million nodes. The result is given in Table 1. In the first column the algorithms are mentioned and in the second column the number of positions solved by each program version is shown. As can be seen, the RPS version is tactically considerably weaker than the unmodified Classic version. However, when augmented with our new enhancements realization probabilities do result in increase tactical strength. This demonstrates clearly the importance of our new enhancements.

In the second set of experiment we tested the playing strength of the programs in actual game play, using the same parameter settings as above. The programs played 600 games against each other, switching sides halfway. They started always from the same 100 positions, and a small randomness was added to avoid repeated games. The thinking time was limited to 10 seconds per move. The results are given in Table 2. The RPS version does perform better than Classic in game play. The added positional understanding gained with a deeper nominal search depth (because of the pruning) does apparently more than compensate for the tactics the program overlooks (because of the pruning). But the new ERPS program version, which benefits from both better positional and tactical understanding, does outperform the other versions with a large margin. The result clearly re-

Table 2. 600-game match results.

Pairings	Score	Winning ratio
RPS vs. Classic	375-225	1.67
ERPS vs. Classic	433-167	2.59
ERPS vs. RPS	372.5-227.5	1.63

veals that ERPS in a genuine improvement that does improve the playing strength of MIA significantly.

5. Conclusions

In this paper we introduced a new variable-depth search scheme in the $\alpha\beta$ search. This scheme is called Enhanced Realization Probability Search (ERPS). It is based on the traditional RPS method which proved to be successful in the game of Shogi and LOA. The novelty is that it limits the search reductions and extensions by considering the rank of the move in the search tree and applying intermediate re-searches. ERPS handles better problems regarding missing tactical moves and spending too much time at bad moves. We showed that our ERPS method solved significantly more LOA endgame positions. Moreover, it was demonstrated that the playing strength of the LOA program MIA is significantly increased by ERPS. Hence, we may conclude that ERPS is a valuable technique augmenting alpha-beta with a more selective tree expansion mechanism such that it improves the winning performance of a program.

Acknowledgments

This research was supported by grants from The Icelandic Centre for Research (RANNÍS) and by a Marie Curie Fellowship of the European Community programme *Structuring the ERA* under contract number MIRG-CT-2005-017284.

References

1. Y. Björnsson, T.A. Marsland, J. Schaeffer, and A. Junghans. Searching with uncertainty cut-offs. *ICCA Journal*, 20(1):29–37, 1997.
2. T. Hashimoto, J. Nagashima, M. Sakuta, J. W. H. M. Uiterwijk, and H. Iida. Automatic realization-probability search. Internal report, Dept. of Computer Science, University of Shizuoka, Hamamatsu, Japan, 2003.
3. R. M. Hyatt. Crafty - chess program. 1996. ftp.cis.uab.edu/pub/hyatt.

4. D. E. Knuth and R. W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.
5. D. Levy, D. Broughton, and M. Taylor. The sex algorithm in computer chess. *ICCA Journal*, 12(1):10–21, 1989.
6. Y. J. Lim and W. S. Lee. Rankcut - a domain independent forward pruning method for games. In *Proceedings of the AAAI 2006*, 2006.
7. T.A. Marsland. Relative efficiency of alpha-beta implementations. In *Proceedings of the 8^{th} International Joint Conference on Artificial Intelligence (IJCAI-83)*, pages 763–766. Karlsruhe, Germany, 1983.
8. A. Reinefeld. An improvement to the Scout search tree algorithm. *ICCA Journal*, 6(4):4–14, 1983.
9. T. Romstad. *An Introduction to Late Move Reductions*. http://www .glaurungchess.com/ lmr.html, 2006.
10. Y. Tsuruoka, D. Yokoyama, and T. Chikayama. Game-tree search algorithm based on realization probability. *ICGA Journal*, 25(3):132–144, 2002.
11. M. H. M. Winands. *Informed Search in Complex Games*. PhD thesis, Universiteit Maastricht, Maastricht, The Netherlands, 2004.