Localizing Search Using Covariance

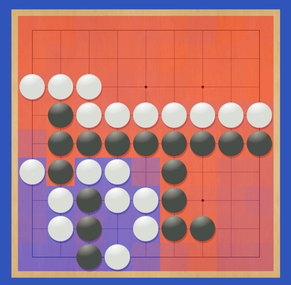
Jason Galbraith, Seth Pellegrino, Peter Drake

Introduction

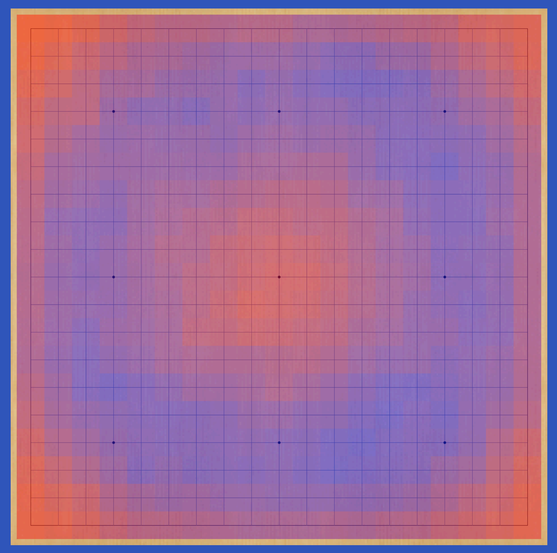
Go is difficult due to the gargantuan size of the game tree. One of the ways of dealing with this is to focus our attention on only the important parts of the tree. Our method for determining which parts of the tree are important is to use covariance.

Covariance

We use the final board state at the end of our monte carlo simulations to collect data on which points seem to influence the outcome of the game. We also look for points that tend to have different end-of-simulation controllers. These two factors combine to create win/control covariance; a point with a high covariance will be one that we can control and will tend to help us win the game.



Covariance on a 9x9 Go problem. Larger circles indicate higher covariance.



Covariance data collected on the empty board. The data suggest that the best places to look for an opening move are in the central ring; this agrees with expert knowledge.

RAVE and Covariance

We combine our standard tree-search algorithm, RAVE, with covariance by pretending moves with higher covariance have had more successful monte carlo simulations. This will tend to make the RAVE algorithm explore these moves more often, biasing the search.

Results

Unfortunately, our results indicate that this naïve method of combining the covariance data with the RAVE player is not significantly helpful.

Conclusions/Future Work

From the covariance maps we can draw, we still feel there is value in pursing better methods of combining this data with the RAVE search. We have identified some issues, and plan to address them.

First, we assume that the best way to control a point is to play directly on it. In Go games, this can quite often not be the case. To correct for this, we can reward any moves that lead to the high covariance points being controlled at the end of a monte carlo simulation.

Secondly, our covariance data is heavily biased by our tree search. If we frequently play a move in the tree search, the covariance of that point will be low. To correct for this, we are considering using a different kind of simulation to avoid bias.

References

Coulom, R. Computing Elo ratings of move patterns in the game of Go. ICGA Journal 30, 198-208 (2007).

Gelly, S. & Silver, D. Combining online and offline knowledge in UCT. In International Conference on Machine Learning, ICML 2007 (2007).

Auer, P., Cesa-Bianchi, N., & Fischer, P. (2002). Finite-time analysis of the multi-armed bandit problem. Machine Learning, 47, 235–256.

Kocsis, L. & Szepesvári, C. Bandit based monte-carlo planning. In ECML-06 (2006).

We'd also like to thank the Willamette Valley REU-RET, Lewis & Clark College, and the National Science Foundation.