Code Review @ Harbrick

Monte Carlo Tree Search

with Reinforcement Learning

Xiyu Xie

RESEARCH HIGHLIGHTS

The Grand Challenge of Computer Go: Monte Carlo Tree Search and Extensions

By Sylvain Gelly, Levente Kocsis, Marc Schoenauer, Michèle Sebag, David Silver, Csaba Szepesvári, Olivier Teytaud

Communications of the ACM, Vol. 55 No. 3, Pages 106-113
10.1145/2093548.2093574

Comments

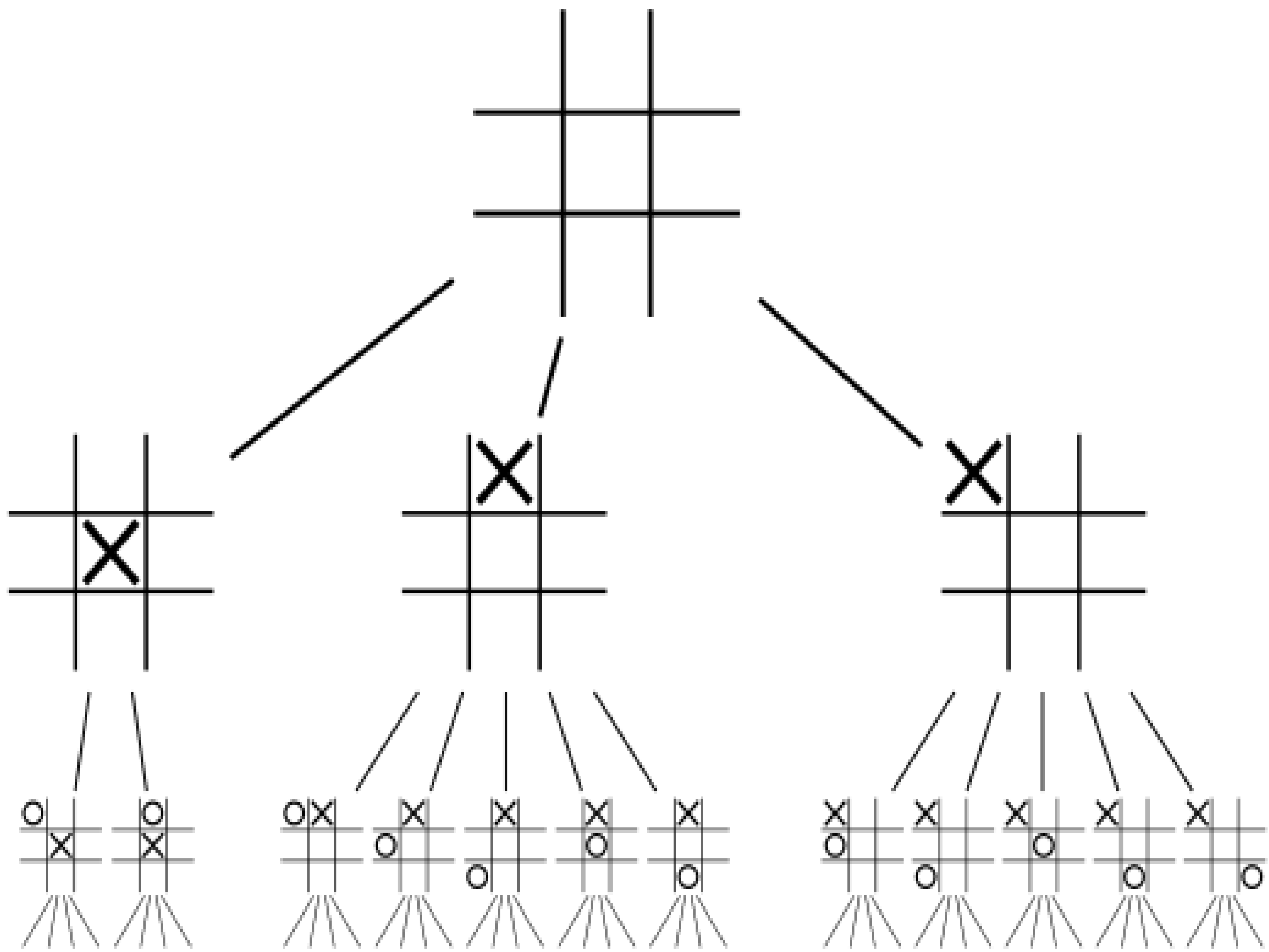
VIEW AS: SHARE:

The ancient oriental game of Go has long been considered a grand challenge for artificial intelligence. For decades, computer Go has defied the classical methods in game tree search that worked so successfully for chess and checkers. However, recent play in computer Go has been transformed by a new paradigm for tree search based on Monte-Carlo methods. Programs based on Monte-Carlo tree search now play at human-master levels and are beginning to challenge top professional players. In this

SIGN IN for Full Access
User Name
Password
» Forgot Password?
» Create an ACM Web Account
SIGN IN

ARTICLE CONTENTS:

Game Tree Search



Simplest Monte Carlo

```
int dummyMove()
{
    //srand (time(NULL));
    int num= rand() %remain;
    int id = findID(num);
    return id;
}

// in select.c
```

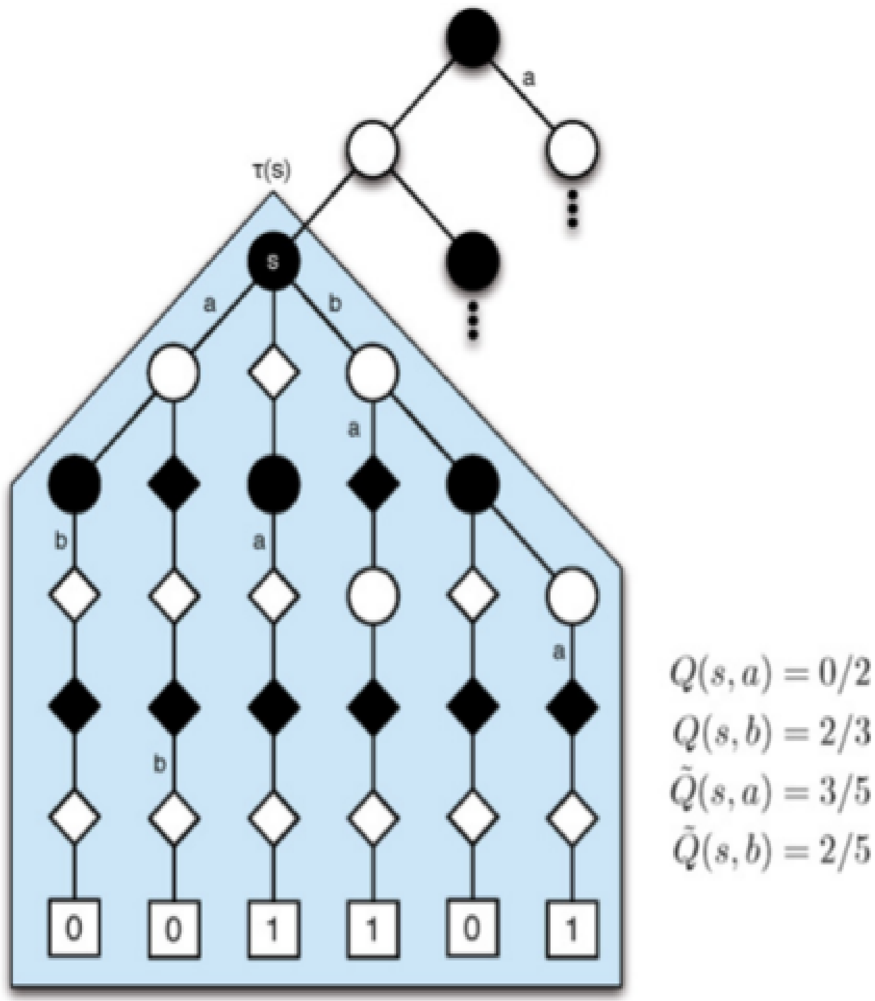
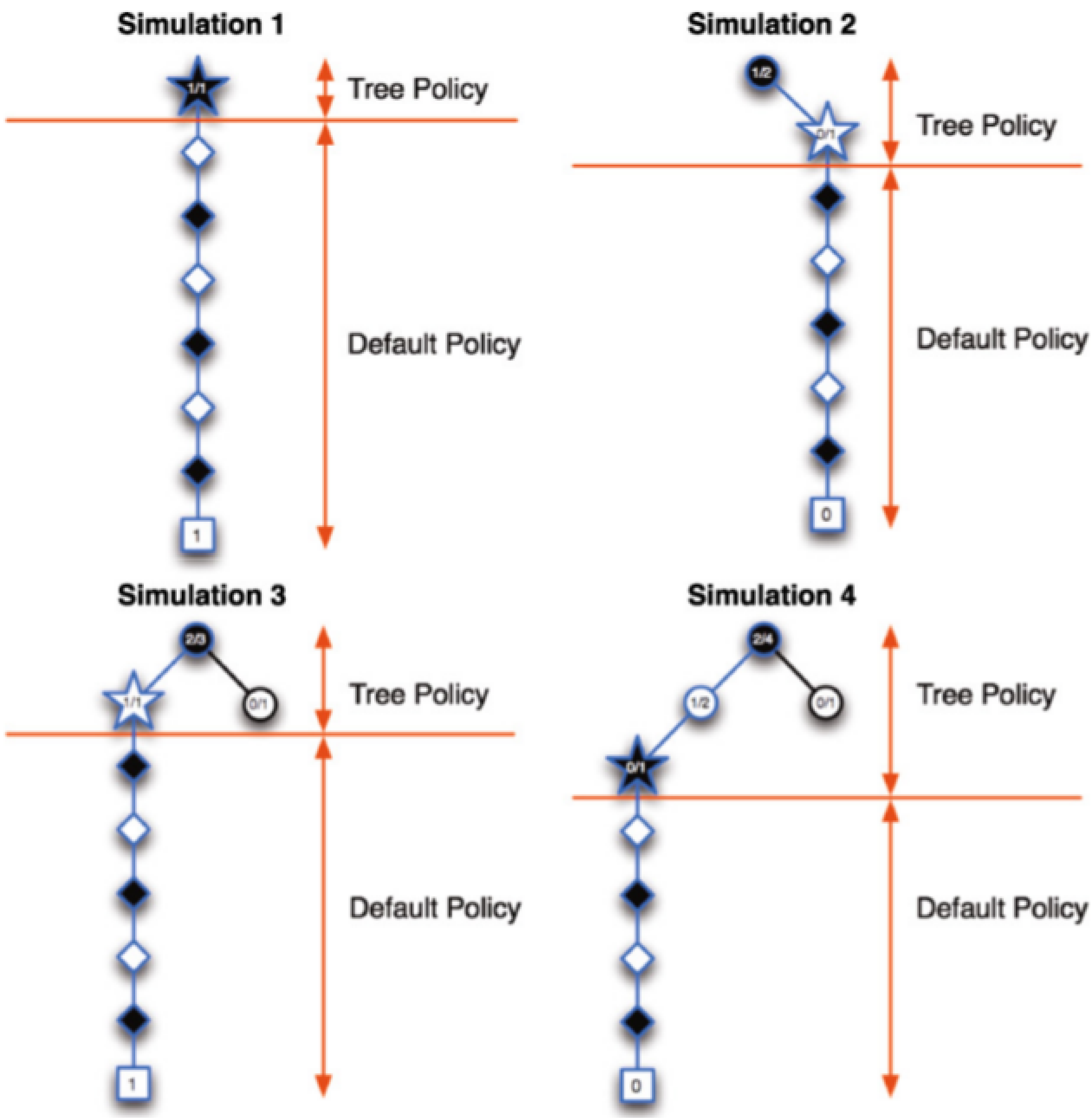


Fig. 4. An example of using the RAVE algorithm to estimate the value of Black moves a and b from state s. Six simulations have been executed from state s, with outcomes shown in the bottom squares. Playing move a immediately led to two losses, and so Monte-Carlo estimation favours move b. However, playing move a at *any* subsequent time led to three wins out of five, and so the RAVE algorithm favours move a. Note that the simulation starting with move a from the root node does not belong to the subtree $\tau(s)$ and does not contribute to the AMAF estimate $\tilde{Q}(s, a)$.


Simulation 1
1/1
Tree Policy
Default Policy
1
Simulation 2
1/2
0/1
Tree Policy
Default Policy
0
Simulation 3
2/3
1/1
0/1
Tree Policy
Default Policy
1
Simulation 4
2/4
1/2
0/1
0/1
Tree Policy
Default Policy
0

Reinforcement Learning

```
float combinedQ(float treeQ, float subQ)
{
      beta *=0.999999;                          // iteratively decreasing
      return (1-beta)*treeQ+beta*subQ;     // mixing two scores
}

// in selection.c
```

$$N(s_t) \leftarrow N(s_t) + 1,$$
$$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1,$$
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \frac{z - Q(s_t, a_t)}{N(s_t, a_t)}.$$

In addition, the AMAF value is updated for every
ree, $s_t \in \mathcal{T}$, and for every subsequent action of the
he AMAF value of (s_t, a_u) is updated according to t

$$\tilde{N}(s_t, a_u) \leftarrow \tilde{N}(s_t, a_u) + 1,$$
$$\tilde{Q}(s_t, a_u) \leftarrow \tilde{Q}(s_t, a_u) + \frac{z - \tilde{Q}(s_t, a_u)}{\tilde{N}(s_t, a_u)}.$$

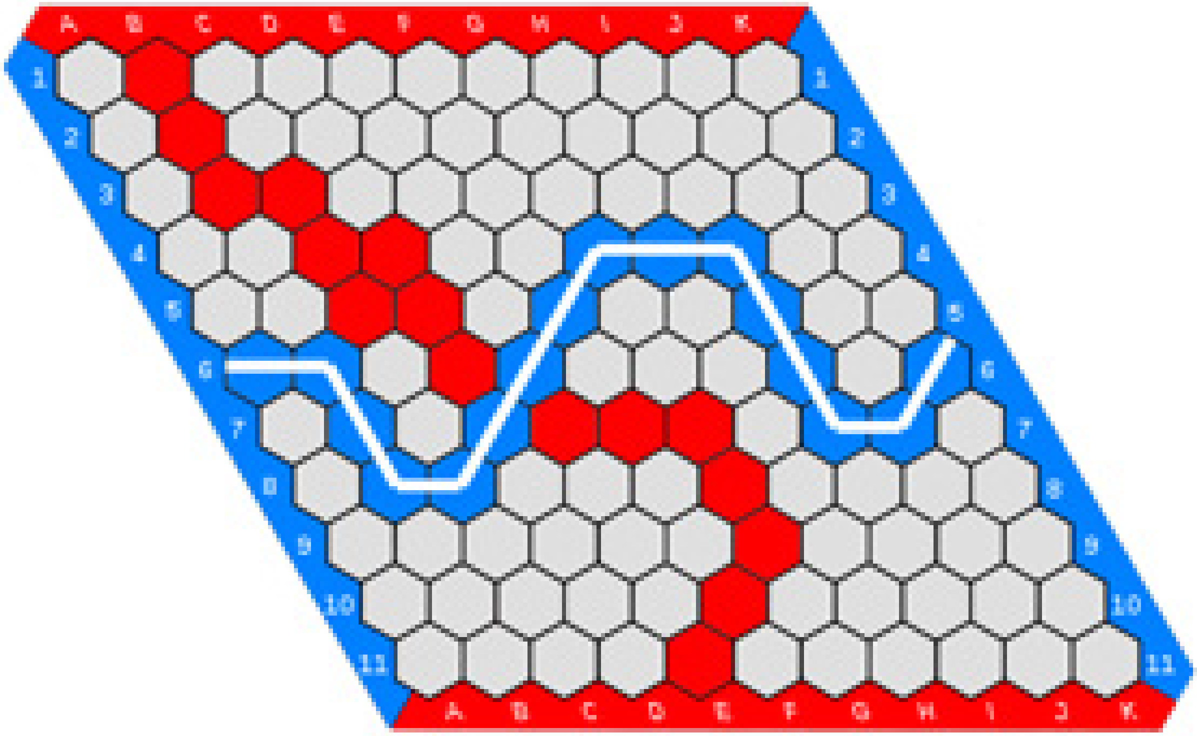
If multiple moves are played at the same inters
rst move at the intersection. If an action a_u is legal
nove.

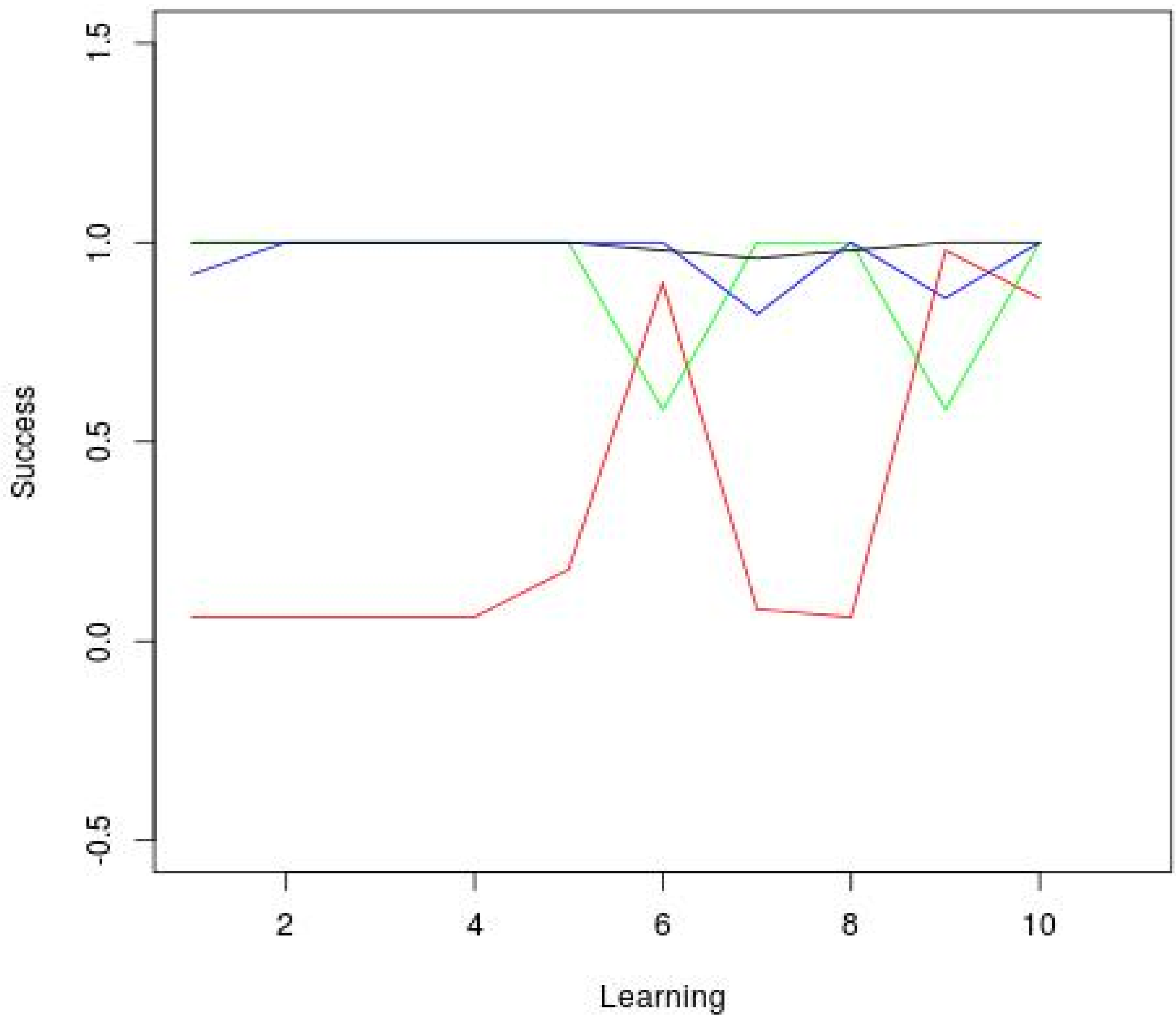
.4. *UCT–RAVE*

The UCT algorithm extends Monte-Carlo tree sea
orating a bonus based on an upper confidence b
ncorporate an exploration bonus,

$$Q_\star^\oplus(s, a) = Q_\star(s, a) + c\sqrt{\frac{\log N(s)}{N(s, a)}}.$$

Winning Score





Success
1.5
1.0
0.5
0.0
-0.5
2
4
6
8
10
Learning