Bounded Suboptimal Game Tree Search

Dor Atzmon and Roni Stern
Ben Gurion University of the Negev
Be'er Sheva, Israel

Abdallah Saffidine
Australian National University
Canberra, Australia

Abstract

....

Bounded Alpha-Beta with multiplicative bound (BAB/m)

A bounded-suboptimal game tree search algorithm with multiplicative bound is defined as a game tree search algorithm that accepts $\gamma \geq 0$ as input and outputs a solution with suboptimality at most $\gamma \cdot V$ from the true minimax value V, where γ is a user-provided parameter. In our case, to adjust BAB to this type of bound, both L and U must be inside these bounds as they are part of the solution, thus: $V - \gamma \cdot V \leq L \leq U \leq V + \gamma \cdot V$, or $1 - \gamma \leq \frac{L}{V} \leq \frac{U}{V} \leq 1 + \gamma$. Algorithm 1 presents the changes needed to be done in BAB in order to return a bounded solution.

....

Worst / best case of BAB

Game plying with BAB

Improvements and comparison (null window search, history heuristic, and more)

Choosing a Bounded-Suboptimal Action

BAB (Algorithm 1) returns a range $[L(n_1), U(n_1)]$ for a given game tree and ϵ. As proven in Theorem ??, the minimax value is in that range, and the size of the range is at most ϵ. Thus, any value chosen in that range can serve as a bounded-suboptimal solution for the given ϵ.

To obtain a concrete policy from the output of BAB the MAX player needs to choose the action that leads to the highest L values. This policy is guaranteed (on expectation) to provide at least a bounded-suboptimal minimax value, for any action the MIN player may choose. By contrast, choosing the action that leads to the highest U values may end up with a strategy whose expected value is smaller than the minimax value by more than ϵ.

ϵ-equilibrium

The optimal policy for MAX and MIN players that use classical Alpha-Beta pruning is known to be a Nash equilibruim, that is, no player can gain by unilaterally deviating from its action. Obviously, a the policy proposed in "" section obtained by BAB in not necessarily a Nash equilibruim. An ϵ-equilibrium (Nisan et al. 2007) is a joint policy in which no player can gain more than ϵ by unilaterally deviating from his strategy. Proving this conjucture is a topic for future research.

Theorem 1. For a game tree G and an ϵ bounded solution, playing the policy proposed in "" results in ϵ-equilibrium

Proof. Assume by the policy in "" child c' is selected.

Case #1: n is a MAX node. By theorem ??, for each child of n

$$U(c) - L(c) \leq \epsilon \tag{1}$$

And

$$V(c) \leq U(c) \tag{2}$$

By the policy in ""

$$\max_{c \in C(n)} L(c) = L(c') \leq V(c') \tag{3}$$

Using (2) and (3) in (1) we get that for each child c of n

$$V(c) - V(c') \leq \epsilon \tag{4}$$

And thus playing the policy proposed in "" results in ϵ-equilibrium □

Related Work

....

Acknowledgements

This research was supported by the Israel Ministry of Science, the Czech Ministry of Education, and by ISF grant #0/17 to Roni Stern.

Algorithm 1: WeightedAB

Input: n - a game tree node
Input: ϵ - an error margin

```
if n is a terminal node then
    return (V(n), V(n))
L(n) ← v_min; U(n) ← v_max
if n is a MAX node then  best_U ← v_min
else if n is a MIN node then  best_L ← v_max
if n is n_1  then  α(n) = v_min; β(n) = v_max
else if p(n) is a MAX node or a MIN node  then
  α(n) = α(p(n)); β(n) = β(p(n))
else
    α(n) ← max(v_min, (α(p(n)) − U(p(n)))/π(n) + U(n))
    β(n) ← min(v_max, (β(p(n)) − L(p(n)))/π(n) + L(n))
foreach child c of n do
    (L(c), U(c)) ← WeightedAB(c, ϵ)
    if n is a MAX node then
        best_U ← max(best_U, U(c))
        L(n) ← max(L(n), L(c))
    else if n is a MIN node then
        best_L ← min(best_L, L(c))
        U(n) ← min(U(n), U(c))
    else
        L(n) ← L(n) + π(c)(L(c) − v_min)
        U(n) ← U(n) − π(c)(v_max − U(c))
    α(n) ← max(L(n), α(n))
    β(n) ← min(U(n), β(n))
    ϵ ← w · min(|α(n)|, |β(n)|)
    if β(n) ≤ α(n) + ϵ and c is not the last child
      of n then  return (L(n), U(n))
if n is a MAX node then  U(n) ← best_U
else if n is a MIN node then  L(n) ← best_L
return (L(n), U(n))
```

References

Nisan, N.; Roughgarden, T.; Tardos, E.; and Vazirani, V. V. 2007. Algorithmic game theory. Cambridge university press.