Computationally Hard Problems

Student name and id: Anders H. Opstrup (s160148)

Collaborator name(s) and id(s):

Hand-in for week: 8

Exercise 1

Consider the scenario underlying the problem GameTreeEvaluation from the lecture, but assume that the tree is a complete quaternary one with 2k levels instead of a binary. Propose a modification of Algorithm 5.30 (randomized evaluation) for this type of game trees. Bound the expected number of leaves evaluated by the algorithm by some value that is lower than the total number of leaves.

Modified randomized evaluation:

$u \leftarrow root$
$result \leftarrow evaluate(v)$

proc $evaluate(v)$
if *v is a leaf* **then**
  return($l(v)$)
else
  let w_1, w_2, w_3 and w_4 be the children of v
  pick one child with probability 1/4 at random; call this a and the rest b, c, d
  $t \leftarrow evaluate(a)$
  if (v is max-node) $\wedge$ ($t == 1$) **then**
    return(0)
  else if (v is min-node) $\wedge$ ($t == 0$) **then**
    return(*evaluate(b)*)
  end if
end if
end proc

Analysis of the algorithm

1. Consider a tree T of depth $2k + 1$ and a min-root.
2. Let $M(T)$ be the expected number of leaves that the modified algorithm looks at when evaluating T.
3. Let $M_{max}(k) = max M(T) | T has depth 2k$.

case 1:

v = min-node
All children have label 0.
A child is picked random with probability of 1/4.
No other child needs to be evaluated, because we already have found a winning move.
$M(T) = 1/4M(T_1) + 1/4M(T_2) + 1/4M(T_3) + 1/4M(T_4) \leq 1/4M_{max}(k) + 1/4M_{max}(k) + 1/4M_{max}(k) + 1/4M_{max}(k) = M_{max}(k)$

case 2:

v = min-node
All children have label 1.
A child is picked random with probability of 1/4.
No matter which child the algorithm picks, the other children needs to be evaluated as well.
$M(T) = M(T_1) + M(T_2) + M(T_3) + M(T_4) \leq 4M_{max}(k)$

case 3:

v = min-node
All children has mixed labels 0 and 1.
A child is picked random with probability of 1/4.
If the algorithm picks a child with label 0, it does not need to evaluate the other children
If the algorithm picks a child with label 1, it needs to evaluate at least one other child
$M(T) = 1/4M(T_1) + 1/4(M(T_2) + M(T_1)) + 1/4(M(T_3) + M(T_2) + M(T_1)) + 1/4(M(T_4) + M(T_3) + M(T_2) + M(T_1)) \leq 2.5M_{max}(k)$

Facts for min-nodes

1. If the label of v is 0, then the time is at most $2.5M_{max}(k)$.
2. If the label of v is 1, then the time is at most $4M_{max}(k)$.

These facts applies for the max-nodes as well just with the obvious changes.

Induction

Consider max-node v which is the root of a tree of depth $2k + 2$
Here we have three cases as well

case 1:

v = max-node
All children have label 0.
A child is picked random with probability of 1/4.
No other child needs to be evaluated, because we already have found a winning move.
No matter which child the algorithm picks, the other children needs to be evaluated as well.
$M(T) \leq 2.5M_{max}(k) + 2.5M_{max}(k) + 2.5M_{max}(k) + 2.5M_{max}(k) = 10M_{max}(k)$

case 2:

v = max-node
All children has mixed labels 0 and 1.
A child is picked random with probability of 1/4.
If the algorithm picks a child with label 1, it does not need to evaluate the other children
If the algorithm picks a child with label 0, it needs to evaluate at least one other child
$M(T) \leq 1/4(4M_{max}(k)+(4+2.5)M_{max}(k))+1/4(4M_{max}(k)+(4+4+2.5)M_{max}(k))+1/4(4M_{max}(k)+(4+4+4+2.5)M_{max}(k)) < 32\ 2/4$

case 3:

v = max-node
All children have label 1.
A child is picked random with probability of 1/4.
No other child needs to be evaluated, because we already have found a winning move.
$M(T) \leq 4M_{max} < 10M_{max}(k)$

Proof

For proof we assume

$$M_{max}(k) \leq 3^k$$

$$M_{max}(k+1) \leq 10M_{max}(k) \leq 10 * 3^k = 30^{k+1}.$$