Zero-Sum Games and Lower Bounds

Abdeselam El-Haman Abdeselam

November 7, 2017

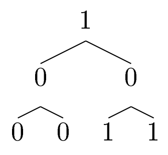
This resolution of the problem given has been tested in 4 different cases, 2 NOR-NOR trees with height 2 and 3 ($k = 1$ and $k = 1.5$) and 2 trees AND-OR with the same heights.

We didn't go further because of the size of the payoff matrix generated by a bigger tree, the runtime of the algorithm became unbearable. Anyway, with the same tree at different heights we can have an intuition in the behaviour of the solution with bigger trees.

For convention in this report, the trees will be represented with sequences, on a by-level order, for example:

$$(1, 0, 0, 0, 0, 1, 1)$$

corresponds to:



And for the vectors, seen the size of these ones, I'll just show the elements e s.t. $e > 0$.

1 Tests

We'll begin with the trees so that $k = 1$. In both cases (NOR-NOR and AND-OR) the result is the same. The positive elements of the algorithm strategies vector are:

1. p(1).- $(Right, Right, Right) \rightarrow 0.5$
2. p(2).- $(Left, Left, Left) \rightarrow 0.5$

And the positive elements of the input strategies vector:

1. q(1).- $(0, 1, 0, 1) \rightarrow 0.5$
2. q(2).- $(1, 0, 1, 0) \rightarrow 0.5$

In this case the opponent (input player) has to give the minimum chances of winning to the algorithm player, so that's why, with the algorithms given by the algorithm player, and the average complexity time of 3 (being the minimum 2 and the maximum 4) so we can say that it's in perfect equilibrium.

$$E[complexity] = \frac{1}{4}q(1)p(1) + \frac{1}{4}q(1)p(2) + \frac{1}{4}q(2)p(1) + \frac{1}{4}q(2)p(2)$$
$$E[complexity] = \frac{2}{4} + \frac{4}{4} + \frac{2}{4} + \frac{4}{4}$$
$$E[complexity] = 3$$

With every other input, the complexity would have been different.

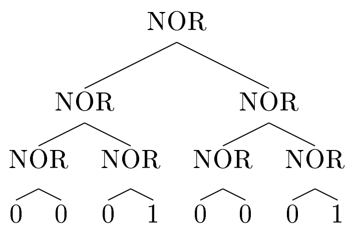
Then we can continue with the tests in bigger trees, for example in the NOR-NOR tree of height 2 the algorithms vector is:

1. p(1).- $(Right, Right, Right, Right, Right, Right, Left) \rightarrow 0.125$
2. p(2).- $(Right, Right, Right, Left, Left, Right, Right) \rightarrow 0.125$
3. p(3).- $(Right, Right, Left, Right, Left, Left, Left) \rightarrow 0.125$
4. p(4).- $(Right, Right, Left, Left, Right, Right, Right) \rightarrow 0.125$
5. p(5).- $(Right, Left, Right, Left, Right, Left, Right) \rightarrow 0.125$
6. p(6).- $(Right, Left, Left, Right, Left, Right, Left) \rightarrow 0.125$
7. p(7).- $(Right, Left, Left, Right, Left, Left, Left) \rightarrow 0.125$
8. p(8).- $(Left, Left, Right, Left, Right, Right, Left) \rightarrow 0.125$

And the input vector:

1. q(1).- $(0, 0, 0, 1, 0, 0, 0, 1) \rightarrow 0.25$
2. q(2).- $(0, 0, 1, 0, 0, 0, 1, 0) \rightarrow 0.25$
3. q(3).- $(0, 1, 0, 0, 1, 0, 0, 0) \rightarrow 0.25$
4. q(4).- $(1, 0, 0, 0, 0, 1, 0, 0) \rightarrow 0.25$

In this case we find a very familiar looking input strategies vector. In fact, these inputs represent every possible permutation capable to convert a 3 level NOR-NOR tree to a 2 level NOR-NOR tree with the same input as seen in the case before. For example, let's take q(1) from our 3 level NOR-NOR tree:



If we evaluate the last level of NOR nodes we can find then:

$$\begin{tikzcd}[column sep=small] & & & \text{NOR} \arrow[dll, dash] \arrow[drr, dash] & & & \\ & \text{NOR} \arrow[dl, dash] \arrow[dr, dash] & & & & \text{NOR} \arrow[dl, dash] \arrow[dr, dash] & \\ 1 & & 0 & & 1 & & 0 \end{tikzcd}$$

Which is q(2) in the input strategies for the 2 level tree of the same kind! And with the same calculus as before, we find an expected complexity of 6, from a minimum of 4 and maximum of 8.

Let's see what happens now when we have an AND-OR:

1. p(1).- $(Right, Right, Right, Right, Right, Right, Left) \rightarrow 0.0961538$
2. p(2).- $(Right, Right, Right, Left, Left, Left, Right) \rightarrow 0.134615$
3. p(3).- $(Right, Right, Left, Right, Left, Right, Left) \rightarrow 0.115385$
4. p(4).- $(Right, Left, Right, Left, Right, Right, Left) \rightarrow 0.115385$
5. p(5).- $(Right, Left, Right, Left, Left, Right, Right) \rightarrow 0.0769231$
6. p(6).- $(Right, Left, Left, Right, Left, Right, Right) \rightarrow 0.0961538$
7. p(7).- $(Right, Left, Left, Right, Right, Left, Left) \rightarrow 0.153846$
8. p(8).- $(Right, Left, Left, Left, Right, Left, Right) \rightarrow 0.0576923$
9. p(9).- $(Left, RIght, Right, Left, Right, Right, Right) \rightarrow 0.0384615$
10. p(10).- $(Left, Right, Right, Left, Right, Left, Left) \rightarrow 0.0384615$
11. p(11).- $(Left, Right, Left, Right, Right, Left, Left) \rightarrow 0.0384615$
12. p(12).- $(Left, Right, Left, Left, Right, Right, Left) \rightarrow 0.0384615$

1. q(1).- $(1, 1, 1, 0, 0, 1, 1, 1) \rightarrow 0.25$
2. q(2).- $(1, 1, 0, 1, 1, 0, 1, 1) \rightarrow 0.25$
3. q(3).- $(1, 0, 1, 1, 1, 1, 0, 1) \rightarrow 0.25$
4. q(4).- $(0, 1, 1, 1, 1, 1, 1, 0) \rightarrow 0.25$

Curiously, we see that this time the input strategies follow a different pattern if we reduce it to a 2-level tree (1,0,0,1 instead of 1,0,1,0). This can be justified because of the more strange probability distribution for the algorithms strategies. In any case, following the same method for the other tree, the expected complexity with these vectors is also 6.

2 Conclusion

We can conclude that the results in a simple game evaluation tree can be propagated in a very larger tree because an optimal game is a game where in every turn the players choose the same strategy, and as the rules of the game don't change while playing, we can consider that the set of best strategies is, in average, the same.