INFO-F413: Project Zero-Sum Games and Lower Bounds

Hilal Rachik 520550

November 2023

Contents

1 Intro

In this report we will discuss the methods and results of Zero-Sum Games application to Game tree evaluation.

2 Method

2.1 Payoff Matrix

An n × m payoff matrix M specifies a two-person zero-sum game. Let the two players be R (for rows) and C (for columns), The value

$$M_{ij} = \text{ payoff due to R by C when R chooses strategy i and C chooses strategy j}$$

Both players follow a discrete probability distribution (p/q) on $\{1, 2, ..., n/m\}$ where R/C chooses strategy i with probability $(p/q)_i$.

2.1.1 Nash Equilibrium

optimal vectors p and q (case when both players choose optimal strategies and can't unilaterally improve their position)

The nash equilibrium can be found by computing both best strategies, this is done, as explained in the paper, by solving the 2 players linear problems using von Neumann's Minimax Theorem:

$$max_p min_j p^T M e_j = min_q max_i e_i^T M q$$

(a player maximise the expected payoffs that have been minimised by other player)

LP:

- For R we use the M^t coefficient because we work with R payoffs due to C actions
- For C we use $-M$ coefficient because matrix encoded with payoff due to R only (inverse values)

for both we add a sum of probability constraint to ensure a valid distribution

2.2 Binary Game NOR-Tree

Game tree evaluation problem: *(Input/Algorithm Analogy)*

the 2 players are the input I (rows), and a set A of deterministic algorithms (columns), the payoff is the running time of the algorithm on the input (number of probes to the Input).

We consider zero-sum games derived from the Game tree evaluation problem in a **binary NOR-tree** of height 2k (complete):

- the set of inputs I, all combinations of size 2^{2k} on {0, 1}, represent the tree leaves.
- the set of deterministic recursive pruning algorithms A, all combinations of size 2^{2k-1} on {0, 1}, associating a child order bit to internal NOR nodes, indicating which of the 2 sub-tree is evaluated first.
- The complexity/payoff is the number of bits of the input (leaves) probed by the algorithm (ordered dfs) (with NOR prunning).

2.3 Matrix From Tree

We have seen both objects Payoff matrix and zero-sum game binary NOR-tree evaluation, now we will see how to generate payoff matrices from the evaluation of binary game NOR-trees.

The binary NOR-tree of height 2K needs 2 parameters to be instantiated and evaluated (see section 2.2):

- Input I (leaves)
- algorithm A (search order)

The matrix will be populated cell by cell with the evaluation of the associated tree(I,A) for every combination of I and A. (So for row i, we evaluated the tree(i, A) for every A.)

The size of the matrix grows exponentially in k, since the number of cells is the size of the Cartesian product of the 2 sets I and A:

$$|I \times A| = 2^{2^{2k}} * 2^{2^{2k}-1} = 2^{2^{2x+1}-1}$$

Matrix size: k=1: (2^7), k=2: (2^{31}), k=3: (2^{127})

3 Results

(**Note**: only results for k=1, k=2 too big)

case k=1:

$$\begin{bmatrix} 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 2 & 2 & 2 & 4 & 3 & 4 & 3 \\ 2 & 2 & 2 & 2 & 3 & 4 & 3 & 4 \\ 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \\ 4 & 4 & 3 & 3 & 2 & 2 & 2 & 2 \\ 4 & 3 & 3 & 2 & 4 & 3 & 3 & 2 \\ 3 & 4 & 2 & 3 & 3 & 4 & 2 & 3 \\ 3 & 3 & 2 & 2 & 3 & 3 & 2 & 2 \\ 3 & 3 & 4 & 4 & 2 & 2 & 2 & 2 \\ 3 & 2 & 4 & 3 & 3 & 2 & 4 & 3 \\ 2 & 3 & 3 & 4 & 2 & 3 & 3 & 4 \\ 2 & 2 & 3 & 3 & 2 & 2 & 3 & 3 \\ 3 & 3 & 3 & 3 & 2 & 2 & 2 & 2 \\ 3 & 2 & 3 & 2 & 3 & 2 & 3 & 2 \\ 2 & 3 & 2 & 3 & 2 & 3 & 2 & 3 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \end{bmatrix}$$

- R player:
  z = 3
  best strategy: [0, 0, 0, 0, 0, $\frac{1}{2}$, 0, 0, 0, 0, $\frac{1}{2}$, 0, 0, 0, 0, 0]
- C player:
  z = -3
  best strategy: [0, 0, $\frac{1}{6}$, $\frac{1}{6}$, $\frac{1}{3}$,$\frac{1}{6}$, 0, $\frac{1}{6}$]

4 Discussion

We can observe that the Minmax Theorem of Von Neumann is respected, and that the optimal expected payoff of the two players are equal (Nash equilibrium)

We will verify the **Game Tree Evaluation results**:
(proofs in syllabus, not relevant to copy paste it)

(1) We know that for any deterministic algorithm, there is an instance of the game tree evaluation problem that forces the algorithm to probe all 4k leaves (no pruning), so in the worst case, a deterministic lower bound is O(n) (n size of the input $2^2k = 4^k$ leaves).

verif: this can be verified in the computed payoff matrix, the highest cell value is $4 = 4k$, linear in the size of the input.

(2) Randomization can improve this lower bound, with an expected number of leaves probed of at most 3^k which makes the randomized lower bound sublinear $\sim O(n^{0.793})$ and thus perform better on average.

verif: this can be verified in the computed payoff matrix, randomly computing the average cell value (over large number of iterations, see code), we obtain $2.62 < 3 = 3k$ (k=1).

Another exact randomized lower bound exists of $O(n^{0.694})$ with each input leaf being set to 1 with probability $p = \frac{3-\sqrt{5}}{2}$, which is not considered in this report/project.